Microsoft Designing and Implementing Microsoft DevOps solutions

Varighed: 5 Days **Kursus Kode: M-AZ400** **Leveringsmetode: Company event (Firmakursus)**

Beskrivelse:

This course provides the knowledge and skills to design and implement DevOps processes and practices. Students will learn how to plan for DevOps, use source control, scale Git for an enterprise, consolidate artifacts, design a dependency management strategy, manage secrets, implement continuous integration, implement a container build strategy, design a release strategy, set up a release management workflow, implement a deployment pattern, and optimize feedback mechanisms.

Firmakursus

Med et firmakursus bliver jeres it-kompetenceudvikling målrettet jeres behov. Det betyder, at vi hjælper med at finde og sammensætte det helt rigtige kursusindhold og den helt rigtige form. Kurset kan afvikles hos os eller kunden, standard eller virtuelt.

Målgruppe:

Students in this course are interested in implementing DevOps processes or in passing the Microsoft Azure DevOps Solutions certification exam.

Agenda:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organization structure
- Describe the benefits of using Source Control
- Migrate from TFVC to Git
- Scale Git for Enterprise DevOps
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Manage application config and secrets
- Develop a project quality strategy
- Plan for secure development practices and compliance rules
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage code quality including: technical debt, SonarCloud, and other tooling solutions
- Manage security policies with open source, OWASP, and
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow

WhiteSource Bolt

- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Integrate and configure ticketing systems with development team's work management
- Implement a mobile DevOps strategy
- Apply infrastructure and configuration as code principles.
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, SaltStack, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Forudsætninger:

Fundamental knowledge about Azure, version control, Agile software development, and core software development principles. It would be helpful to have experience in an organization that delivers software.

Test og certificering

Indhold:

Module 1: Planning for DevOps

- Transformation Planning
- Project Selection
- Team Structures
- Migrating to Azure DevOps

Lab: Agile Planning and Portfolio Management with Azure Boards

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package

Module 6: Managing Application Config and Secrets

- Introduction to Security
- Implement secure and compliant development process
- Rethinking application config data
- Manage secrets, tokens, and certificates
- Implement tools for managing security and compliance in a pipeline

Lab : Integrating Azure Key Vault with Azure DevOps

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including

Module 13: Implement process for routing system feedback to development teams

- Implement Tools to Track System Usage, Feature Usage, and Flow
- Implement Routing for Mobile Application Crash Report Data
- Develop Monitoring and Status Dashboards
- Integrate and Configure Ticketing SystemsLab : Monitoring Application Performance

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker

security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM

how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for

- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems

Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 2: Getting started with Source Control

- What is Source Control
- Benefits of Source Control
- Types of Source Control Systems
- Introduction to Azure Repos
- Introduction to GitHub
- Migrating from Team Foundation Version Control (TFVC) to Git in Azure Repos
- Authenticating to Git in Azure Repos

Lab : Version Controlling with Git

After completing this module, students will be able to:

- Describe the benefits of using Source Control
- Describe Azure Repos and GitHub
- Migrate from TFVC to Git

client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 7: Managing Code Quality and Security Policies

- Managing Code Quality
- Managing Security Policies Lab :

with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end

Module 3: Scaling Git for enterprise DevOps

- How to Structure your Git Repo
- Git Branching Workflows
- Collaborating with Pull Requests in Azure Repos
- Why care about GitHooks
- Fostering Inner Source

Lab : Code Review with Pull Requests

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package

Managing Technical Debt with Azure DevOps and SonarCloud

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages

traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation

security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation ToolsLab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and

Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 4: Consolidating Artifacts ; Designing a Dependency Management Strategy

- Packaging Dependencies
- Package Management
- Migrating and Consolidating Artifacts

Lab : Updating Packages

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g.

Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 8: Implementing a Container Build Strategy

- Implementing a Container Build StrategyLab : Modernizing Existing ASP.NET Apps with Azure

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)

Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 15: Azure Deployment Models and Services

- Deployment Modules and Options
- Azure Infrastructure-as-a-Service (IaaS) Services
- Azure Platform-as-a-Service (PaaS) services
- Serverless and HPC Computer Services
- Azure Service FabricLab : Deploying a Dockerized Java app to Azure Web App for Containers

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices

VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job

- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies

infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 5: Implementing Continuous Integration with Azure Pipelines

- The concept of pipelines in DevOps
- Azure Pipelines
- Evaluate use of Hosted vs Private Agents
- Agent Pools
- Pipelines and Concurrency
- Azure DevOps and Open Source Projects (Public Projects)
- Azure Pipelines YAML vs Visual Designer
- Continuous Integration Overview
- Implementing a Build Strategy
- Integration with Azure Pipelines
- Integrate External Source Control with Azure Pipelines
- Set Up Private Agents
- Analyze and Integrate Docker Multi-Stage Builds

Lab : Enabling Continuous Integration with Azure Pipelines

Lab : Integrating External Source Control with Azure Pipelines

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 9: Manage Artifact versioning, security ; compliance

- Package security
- Open source software
- Integrating license and vulnerability scans
- Implement a versioning strategyLab : Manage Open Source Security and License with WhiteSource

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 16: Create and Manage Kubernetes Service Infrastructure

- Azure Kubernetes ServiceLab : Deploying a multi-container application to Azure Kubernetes Service

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers

on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available

- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling

- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow

deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 17: Third Party Infrastructure as Code Tools available with Azure

- Chef
- Puppet
- Ansible
- Terraform

- Lab : Configuring Pipelines as Code with YAML

user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 10: Design a Release Strategy

- Introduction to Continuous Delivery
- Release strategy recommendations
- Building a High-Quality Release pipeline
- Choosing a deployment pattern
- Choosing the right release management tool

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration
- Lab : Setting up secrets in the pipeline with Azure Key vault
- Lab : Setting up and Running Functional Tests
- Lab : Using Azure Monitor as release gate
- Lab : Creating a release Dashboard
- Lab : Infrastructure as Code
- Lab : Automating Your Infrastructure Deployments in the Cloud with Terraform and Azure Pipelines

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package

matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using

security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration

a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation ToolsLab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools

Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation ToolsLab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 18: Implement Compliance and Security in your Infrastructure

- Security and Compliance Principles with DevOps
- Azure security CenterLab : Implement Security and Compliance in an Azure DevOps Pipeline

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational

- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 11: Set up a Release Management Workflow

- Create a Release Pipeline
- Provision and Configure Environments
- Manage and Modularize Tasks and Templates
- Integrate Secrets with the release pipeline
- Configure Automated Integration and Functional Test Automation
- Automate Inspection of Health

- Lab : Configuring Pipelines as Code with YAML
- Lab : Setting up secrets in the pipeline with Azure Key vault
- Lab : Setting up and Running Functional Tests
- Lab : Using Azure Monitor as release gate
- Lab : Creating a release Dashboard
- Lab : Infrastructure as Code
- Lab : Automating Your Infrastructure Deployments in the Cloud with Terraform and Azure Pipelines

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable

structure

- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release

sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need

process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation ToolsLab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure

multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation ToolsLab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 19: Recommend and design system feedback mechanisms

- The inner loop
- Continuous Experimentation mindset
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback
- Design process to automate application analyticsLab : Integration between Azure DevOps and Teams

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews

user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 12: Implement an appropriate deployment pattern

- Introduction to Deployment Patterns
- Implement Blue Green Deployment
- Feature Toggles
- Canary Releases
- Dark Launching
- AB Testing
- Progressive Exposure DeploymentLab : Feature Flag Management with Launch Darkly and Azure DevOps

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and

Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline

Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation ToolsLab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze

- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation ToolsLab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Module 20: Optimize feedback mechanisms

- Site Reliability Engineering
- Analyze telemetry to establish a baseline
- Perform ongoing tuning to reduce meaningless or non-actionable alerts
- Analyze alerts to establish a baseline
- Blameless Retrospectives and a Just Culture

After completing this module, students will be able to:

- Plan for the transformation with shared goals and timelines
- Select a project and identify project metrics and KPIs
- Create a team and agile organizational structure
- Design a tool integration strategy
- Design a license management strategy (e.g. VSTS users)
- Design a strategy for end-to-end traceability from work items to working software
- Design an authentication and access strategy
- Design a strategy for integrating on-premises and cloud resources
- Explain how to structure Git repos
- Describe Git branching workflows
- Leverage pull requests for collaboration and code reviews
- Leverage Git hooks for automation
- Use git to foster inner source across the organization
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Migrate and consolidate artifacts
- Migrate and integrate source control measures
- Implement and manage build infrastructure
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Manage application config and secrets
- Implement tools for managing security and compliance in pipeline
- Manage code quality including: technical

- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

debt SonarCloud, and other tooling solutions
- Manage security policies with open source and OWASP
- Implement a container strategy including how containers are different from virtual machines and how microservices use containers
- Implement containers using Docker
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating
- Configure secure access to package feeds
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Differentiate between a release and a deployment
- Define the components of a release pipeline
- Explain things to consider when designing your release strategy
- Classify a release versus a release process and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue, and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure

Deployment
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work managementModule 14: Infrastructure and Configuration Azure Tools
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Desired State Configuration (DSC)
- Azure Automation with DevOps
- Additional Automation Tools Lab : Azure Deployments using Resource Manager Templates
- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Flere Informationer:

For yderligere informationer eller booking af kursus, kontakt os på tlf.nr.: 44 88 18 00

training@globalknowledge.dk

www.globalknowledge.com/da-dk/

Global Knowledge, Stamholmen 110, 2650 Hvidovre