Enterprise Computing Solutions - Education Services

TRAINING OFFERING

Du kan nå oss her

Postboks 6562 ETTERSTAD, 0606 Oslo, Norge

Email: kurs.ecs.no@arrow.com
Phone: +47 22 02 81 00

Designing and Implementing Microsoft DevOps Solutions

CODE:	LENGTH:	PRICE:
MCS_AZ-400T00	40 Hours (5 days)	kr29,300.00

Description

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

Objectives

After completing this course, 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 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 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
•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 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

Audience

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

Prerequisites

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.

Programme

Lessons
•Transformation Planning
•Project Selection
•Team Structures
Module 1: Planning for DevOps •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 Module 2: Getting started with Source Control
Lessons
•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 Module 3: Scaling Git for enterprise DevOps
Lessons
•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:
•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

Lessons
•Packaging Dependencies
•Package Management
Module 4: Consolidating Artifacts & Designing a Dependency Management Strategy •Migrating and Consolidating Artifacts

After completing this module, students will be able to:
•Recommend artifact management tools and practices
•Abstract common packages to enable sharing and reuse
•Migrate and consolidate artifacts
Lab : Updating Packages •Migrate and integrate source control measures

Lessons
•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
Module 5: Implementing Continuous Integration with Azure Pipelines •Analyze and Integrate Docker Multi-Stage Builds
Lab : Enabling Continuous Integration with Azure Pipelines Lab : Integrating External Source Control with Azure Pipelines
Lab : Integrate Jenkins with Azure Pipelines Lab : Deploying a Multi-Container Application

After completing this module, students will be able to: • Implement and manage build infrastructure
• Explain why continuous integration matters • Implement continuous integration using Azure DevOps

Lessons
•Introduction to Security
•Implement secure and compliant development process
•Rethinking application config data
•Manage secrets, tokens, and certificates
Module 6: Managing Application Config and Secrets •Implement tools for managing security and compliance in a pipeline

After completing this module, students will be able to:
Lab : Integrating Azure Key Vault with Azure DevOps • Manage application config and secrets

Lessons
•Managing Code Quality
Module 7: Managing Code Quality and Security Policies •Managing Security Policies
Lab : Managing Technical Debt with Azure DevOps and SonarCloud

After completing this module, students will be able to:
•Manage code quality including: technical debt SonarCloud, and other tooling solutions
•Manage security policies with open source and OWASP

Lessons
Module 8: Implementing a Container Build Strategy •Implementing a Container Build Strategy
Lab : Modernizing Existing ASP.NET Apps with Azure

After completing this module, students will be able to:
•Implement a container strategy including how containers are different from virtual machines and how microservices use containers
•Implement containers using Docker

Lessons
•Package security
•Open source software
•Integrating license and vulnerability scans

Module 9: Manage Artifact versioning, security & compliance •Implement a versioning strategy (git version)
Lab : Manage Open Source Security and License with WhiteSource

After completing this module, students will be able to:
•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

Lessons
•Introduction to Continuous Delivery
•Release strategy recommendations
•Building a High-Quality Release pipeline
•Choosing a deployment pattern

Module 10: Design a Release Strategy •Choosing the right release management tool

After completing this module, students will be able to:
•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

Lessons
•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

Module 11: Set up a Release Management Workflow •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

After completing this module, students will be able to:
•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

Lessons
•Introduction to Deployment Patterns
•Implement Blue Green Deployment
•Feature Toggles
•Canary Releases
•Dark Launching
•AB Testing

Module 12: Implement an appropriate deployment pattern •Progressive Exposure Deployment

After completing this module, students will be able to:
•Describe deployment patterns
•Implement Blue Green Deployment
•Implement Canary Release

Lab : Feature Flag Management with LaunchDarkly and Azure DevOps •Implement Progressive Exposure Deployment

Module 13: Implement process for routing system feedback to development teams
Lessons
•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 Systems Lab : Monitoring Application Performance

After completing this module, students will be able to:
•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 management
Lessons
•Introduction to Mobile DevOps
•Introduction to Visual Studio App Center
•Manage mobile target device sets and distribution groups
•Manage target UI test device sets
•Provision tester devices for deployment
Module 14: Implement a mobile DevOps strategy •Create public and private distribution groups

After completing this module, students will be able to:
•Implement a mobile DevOps strategy Module 15: Infrastructure and Configuration Azure Tools
Lessons
•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

After completing this module, students will be able to:
•Apply infrastructure and configuration as code principles
•Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
Lessons
•Deployment Modules and Options
•Azure Infrastructure-as-a-Service (IaaS) Services
•Azure Platform-as-a-Service (PaaS) services
•Serverless and HPC Computer Services
Module 16: Azure Deployment Models and Services •Azure Service Fabric

After completing this module, students will be able to:
Lab : Azure Automation - IaaS or PaaS deployment •Describe deployment models and services that are available with Azure
Lessons
Module 17: Create and Manage Kubernetes Service Infrastructure •Azure Kubernetes Service

After completing this module, students will be able to:
Lab : Deploying a multi-container application to Azure Kubernetes Service •Deploy and configure a Managed Kubernetes cluster
Lessons
•Chef
•Puppet
•Ansible
Module 18: Third Party Infrastructure as Code Tools available with Azure •Terraform 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:
•Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, SaltStack, and Terraform
Lessons
•Security and Compliance Principles with DevOps
Module 19: Implement Compliance and Security in your Infrastructure •Azure security Center
Lab : Implement Security and Compliance in an Azure DevOps Pipeline

After completing this module, students will be able to:
•Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
•Implement compliance and security in your application infrastructure

Lessons
•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 analytics

Module 20: Recommend and design system feedback mechanisms
Lab : Integration between Azure DevOps and Teams

After completing this module, students will be able to:
•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

Module 21: Optimize feedback mechanisms
Lessons
•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:
•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

Session Dates

Ved forespørsel. Vennligst kontakt oss

Tilleggsinformasjon

Denne treningen er også tilgjengelig som trening på stedet. Kontakt oss for å finne ut mer.