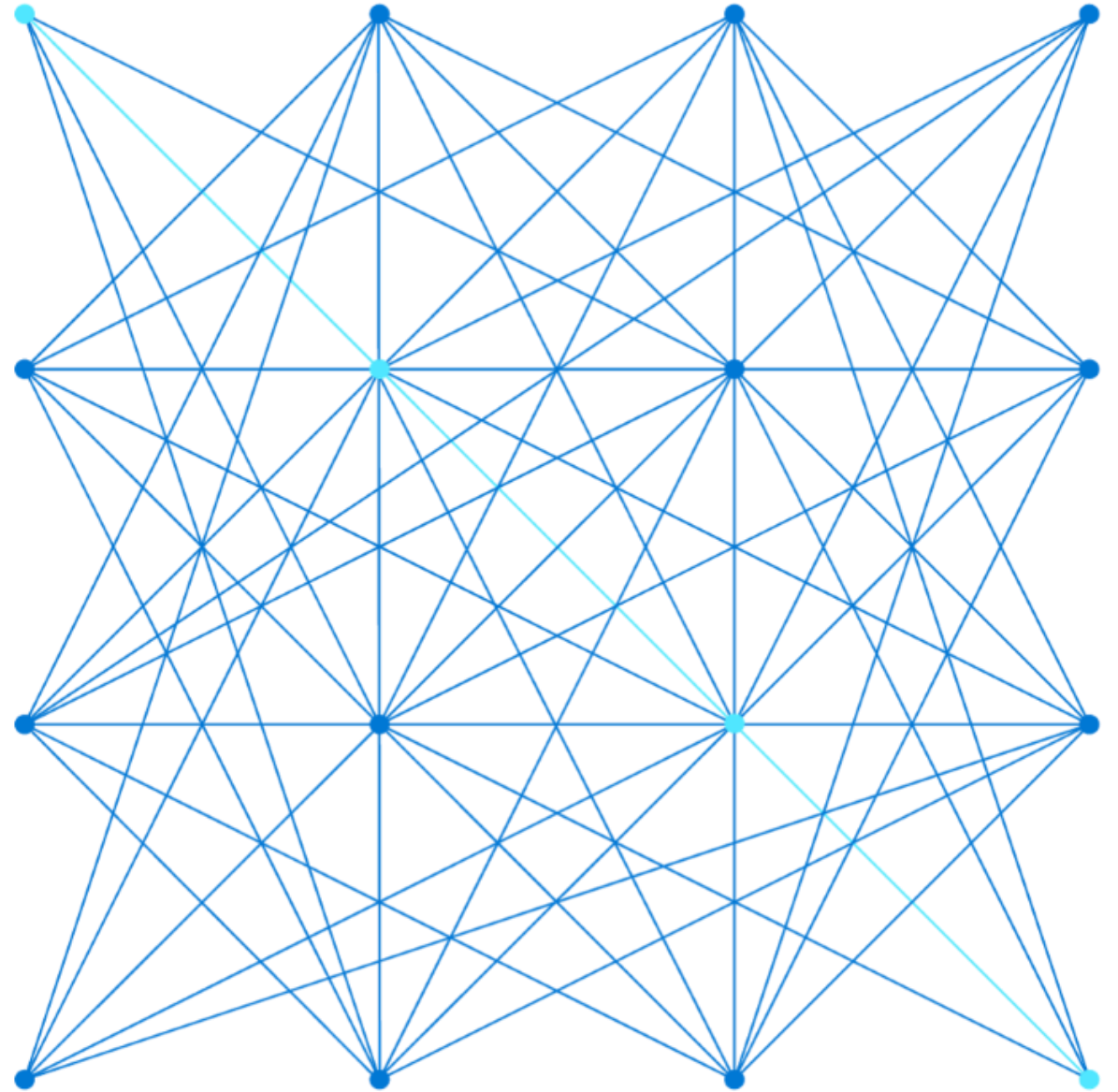


AZ-400.00

Module 11: Implementing Continuous Deployment using Azure Pipelines



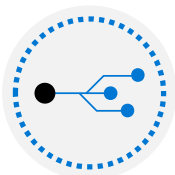
Lesson 01: Module overview



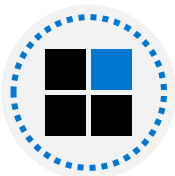
Module overview



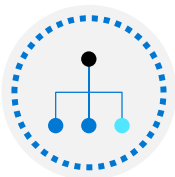
Lesson 1: Module overview



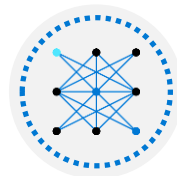
Lesson 2: Create a release pipeline



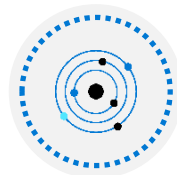
Lesson 3: Provision and configure environments



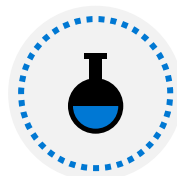
Lesson 4: Manage and modularize tasks and templates



Lesson 5: Configure automated integration and functional test automation



Lesson 6: Automate inspection of health



Lesson 7: Labs



Lesson 8: Module review and takeaways

Learning objectives

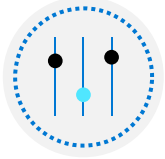
After completing this module, students will be able to:



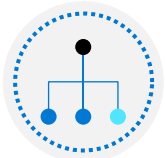
Explain the terminology used in Azure DevOps and other release management tooling



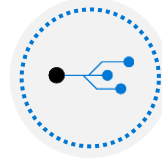
Describe what a build and release task is, what it can do, and some available deployment tasks



Explain why you sometimes need multiple release jobs in one release pipeline



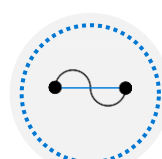
Differentiate between multi-agent and multi-configuration release job



Use release variables and stage variables in your release pipeline

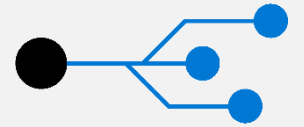


Deploy to environment securely using a service connection

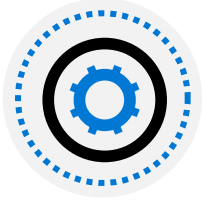


List the different ways to inspect the health of your pipeline and release by using alerts, service hooks, and reports

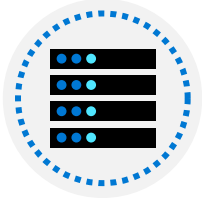
Lesson 02: Create a release pipeline



Azure DevOps release pipelines



Support pipelines as code (via YAML)



Most capabilities from classic release pipelines are available

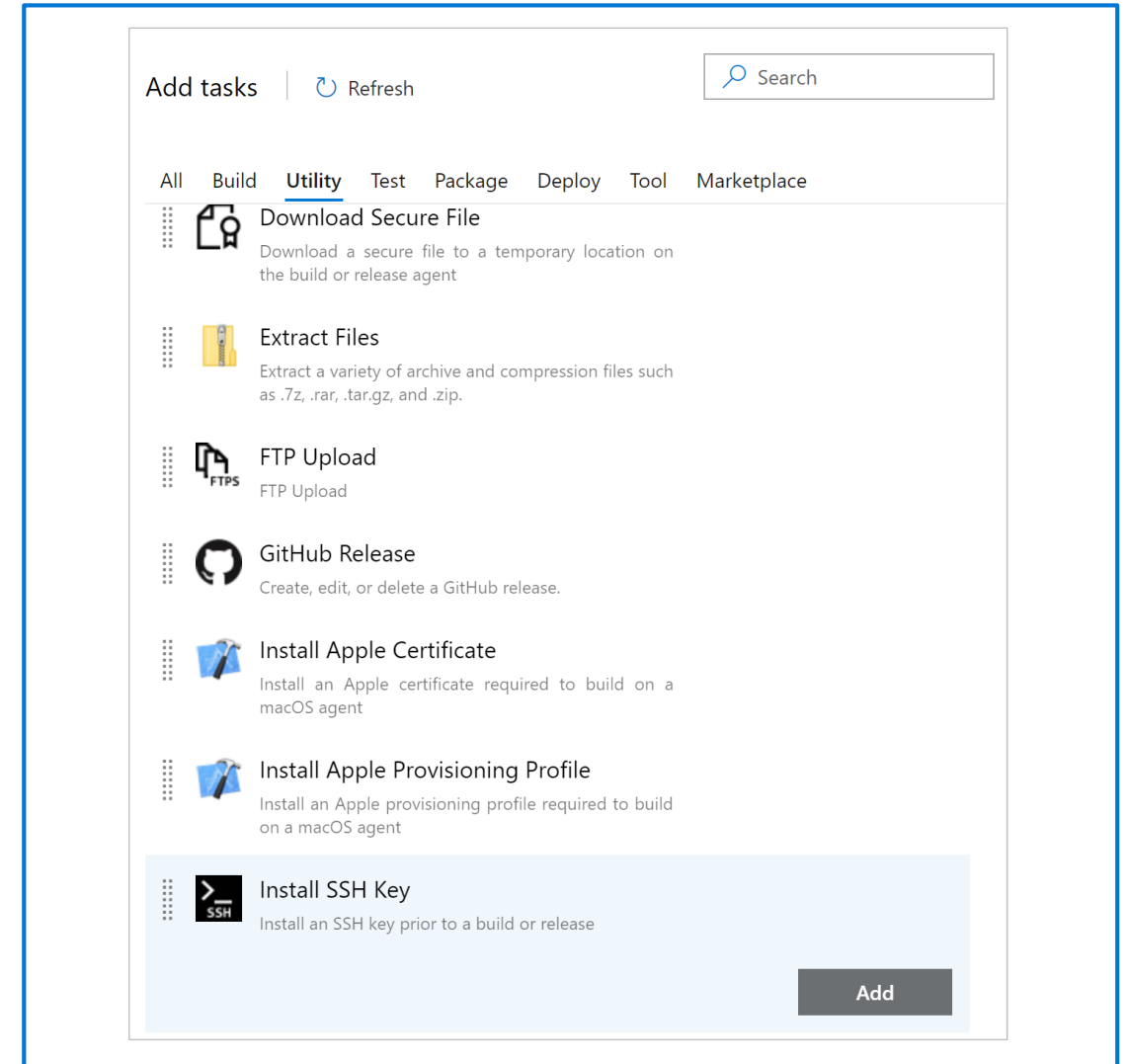
Build and release tasks

Units of executable code used to perform designated actions in a specified order

Tasks building, testing, running utilities, packaging, and deploying

Extensible model

Community tasks available in marketplace



The screenshot displays the 'Utility' tab in the Azure DevOps Marketplace. At the top, there are links for 'Add tasks' and 'Refresh', along with a search bar. Below the navigation bar, a list of tasks is shown, each with an icon, a title, and a brief description. The tasks listed are: 'Download Secure File' (download a secure file to a temporary location), 'Extract Files' (extract archive and compression files), 'FTP Upload' (FTP Upload), 'GitHub Release' (create, edit, or delete a GitHub release), 'Install Apple Certificate' (install an Apple certificate required to build on a macOS agent), 'Install Apple Provisioning Profile' (install an Apple provisioning profile required to build on a macOS agent), and 'Install SSH Key' (install an SSH key prior to a build or release). The 'Install SSH Key' task is highlighted with a blue background and an 'Add' button at the bottom right.

Add tasks | Refresh

Search

All Build Utility Test Package Deploy Tool Marketplace

-  **Download Secure File**
Download a secure file to a temporary location on the build or release agent
-  **Extract Files**
Extract a variety of archive and compression files such as .7z, .rar, .tar.gz, and .zip.
-  **FTP Upload**
FTP Upload
-  **GitHub Release**
Create, edit, or delete a GitHub release.
-  **Install Apple Certificate**
Install an Apple certificate required to build on a macOS agent
-  **Install Apple Provisioning Profile**
Install an Apple provisioning profile required to build on a macOS agent
-  **Install SSH Key**
Install an SSH key prior to a build or release

Add

Release jobs

A job is a series of tasks that run sequentially on the same set of targets

Can be combined in one pipeline to enable multi-platform deployment:

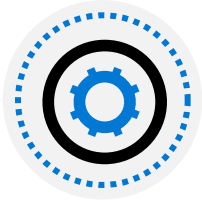
E.g., deploy .NET backend via Windows, iOS app via MacOS and Angular frontend via Linux

Jobs run on the host machine where the agent is installed

The screenshot displays the 'Release Jobs Picture' pipeline in Azure DevOps. The interface includes a breadcrumb 'All pipelines > Release Jobs Picture' and a tabbed menu with 'Pipeline', 'Tasks', 'Variables', 'Retention', 'Options', and 'History'. The 'Tasks' tab is active, showing a list of jobs under the 'Development' deployment process. The jobs are organized into three groups: 'macOS Job' (Run on agent), 'Deployment group job' (Run on deployment group), and 'Ubuntu Job' (Run on agent). Each group contains specific tasks with their respective icons and names.

Job Name	Run On	Tasks
macOS Job	Run on agent	Publish to the App Store TestFlight track (Apple App Store Release)
Deployment group job	Run on deployment group	- PowerShell Script (PowerShell) - Deploy IIS Website/App: (IIS Web App Deploy)
Ubuntu Job	Run on agent	Release n/a to internal (Google Play - Release)

Multi-configuration and multi-agent

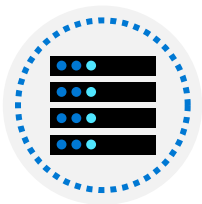


Multi-configuration: Run the same set of tasks on multiple configurations:

Run the release once with configuration setting A on WebApp A and setting B for WebApp B

Deploy to different geographic regions

D-mw-multiconfiguration



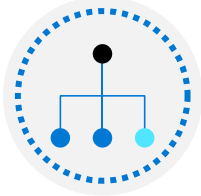
Multi-agent: Run the same set of tasks on multiple agents using the specified number of agents:

Deploy same bits to a farm of servers

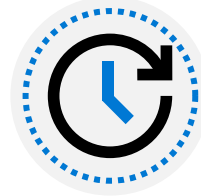
Thoughts on: How to use release jobs

Do you see a purpose for release jobs in your pipeline? How would you set it up?

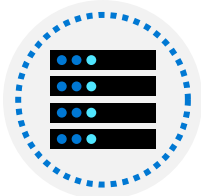
Topics you might want to consider are:



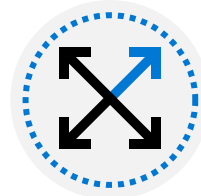
Do you have artifacts from multiple sources?



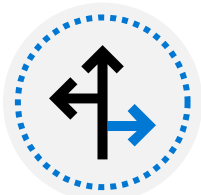
How long does your release take?



Do you want to run deployments on different servers simultaneously?

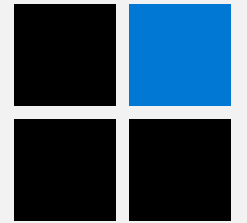


Can you run your deployment in parallel or does it need to run in sequence?

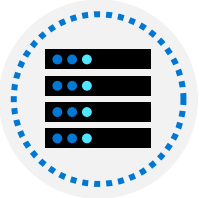


Do you need multiple platforms?

Lesson 03: Provision and configure environments



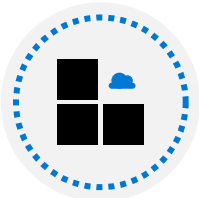
Provision and configure **target** environments



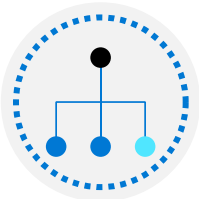
On-premises servers



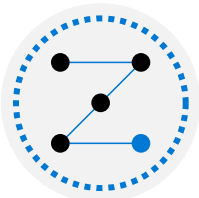
Cloud servers or Infrastructure as a Service (IaaS). For example, virtual machines or networks



Platform as a Service (PaaS) and Functions as a Service (FaaS). For example, Azure SQL Database in both PaaS and Serverless options

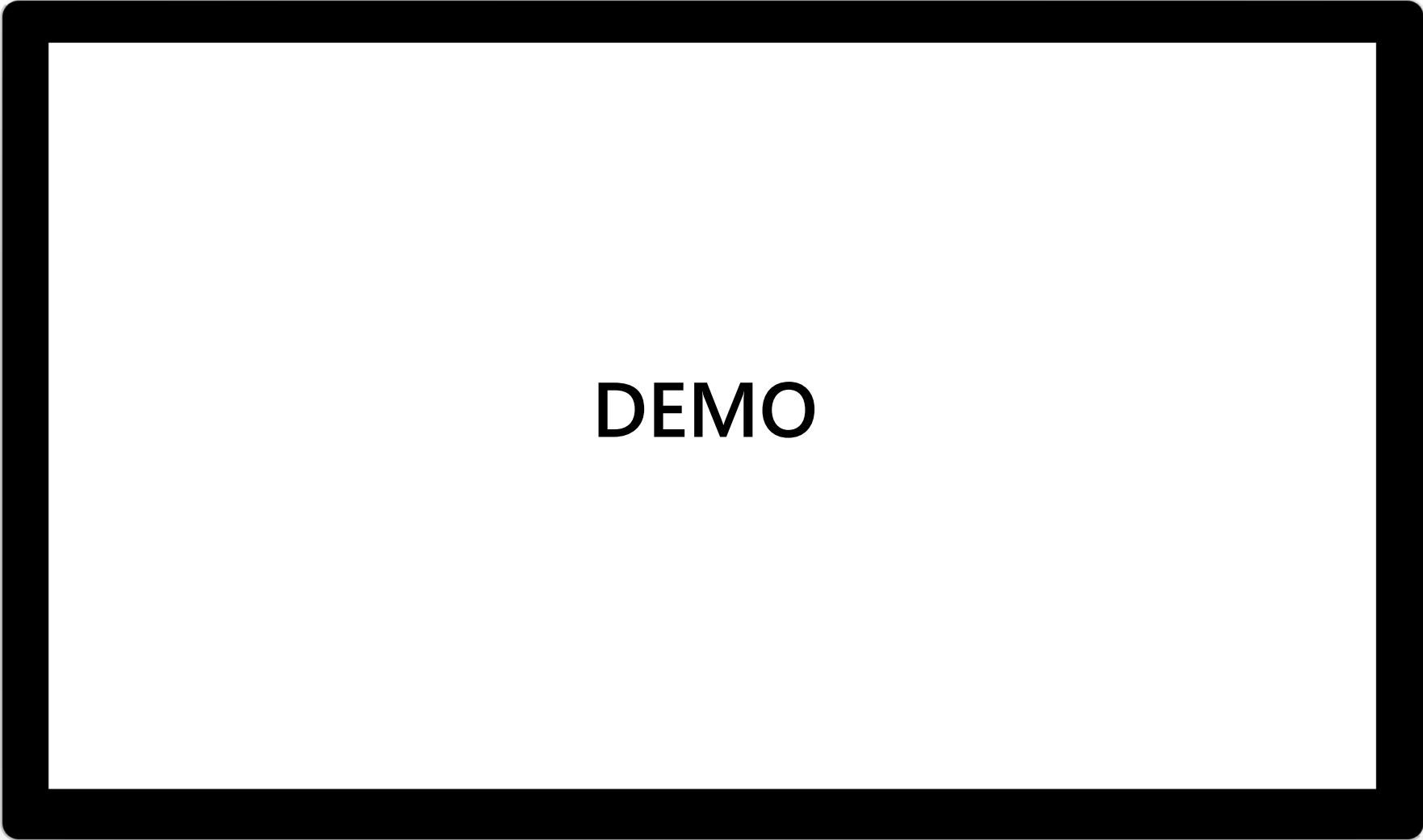


Clusters



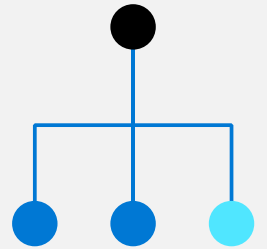
Service connections

Demonstration: Setting up **service connections**



DEMO

Lesson 04: Manage and modularize tasks and templates



Task groups

Encapsulate a sequence of tasks in one reusable task

Parameterization of task groups to make reuse easier

Standardize and centrally manage deployment steps for all your applications

The screenshot shows the 'Task groups > Demo' configuration page in Azure DevOps. The left sidebar lists 'Demo' (Version 1.0) with a plus icon. The main area shows the configuration for the 'Demo' task group. It includes a 'Version' dropdown set to '1.0', a 'Name' field with 'Demo', a 'Description' field, a 'Category' dropdown set to 'Deploy', and a 'Parameters' section. The parameters section has a table with columns 'Name', 'Default value', and 'Description'.

Name	Default value	Description
HostingPlan	DemoHostingPlan	
ResourceGroupName	mcourse-cd	Provide the name of a res...
ServerName	mcoursecdserv	
slot		Enter or Select an existin...
WebsiteName	mcoursecdserv	

The screenshot shows the 'Add tasks' dialog in Azure DevOps. It has a search bar with 'demo' entered, a 'Refresh' button, and a list of tasks. The 'Demo' task is highlighted, showing its icon and name.

Note: Task groups are not currently supported in YAML. Use templates instead.

Variables in release pipelines

Predefined Variables

Pipeline variables

Stage Variables

Variable groups

Normal and Secret variables

Variables Retention Options History

Filter by keywords Scope

Name	Value	Scope
Prefix	Demo	Release
ServerName	DevServer	Dev
ServerName	TestServer	Test
Password	*****	Release

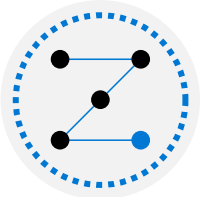
D-mw-variables

Variable groups

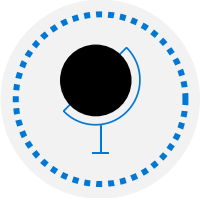
A variable group is used to store values that you want to make available across multiple builds and release pipelines:



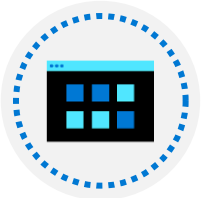
Store the username and password for a shared server



Store a share connection string



Store the geolocation of an application



Store all settings for a specific application

Custom build/release tasks (#mw. i.e. you can create your own tasks, see [here](#))

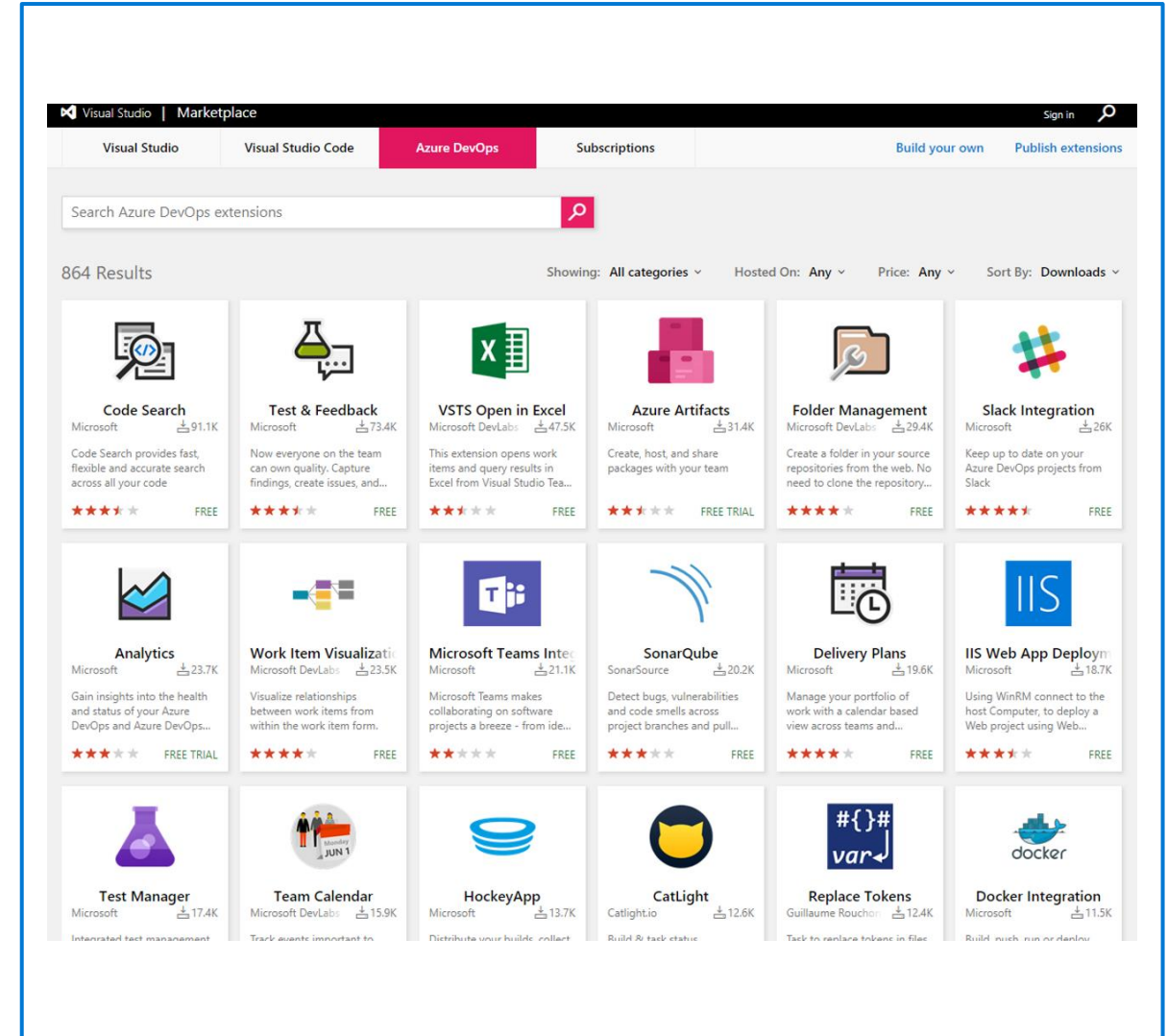
Private or public accessible

Access to variables that are otherwise not accessible

Use and reuse secure endpoint to a target server

Safely and efficiently distribute across your whole organization

Users do not see implementation details



Lesson 05: Configure automated integration and functional test automation

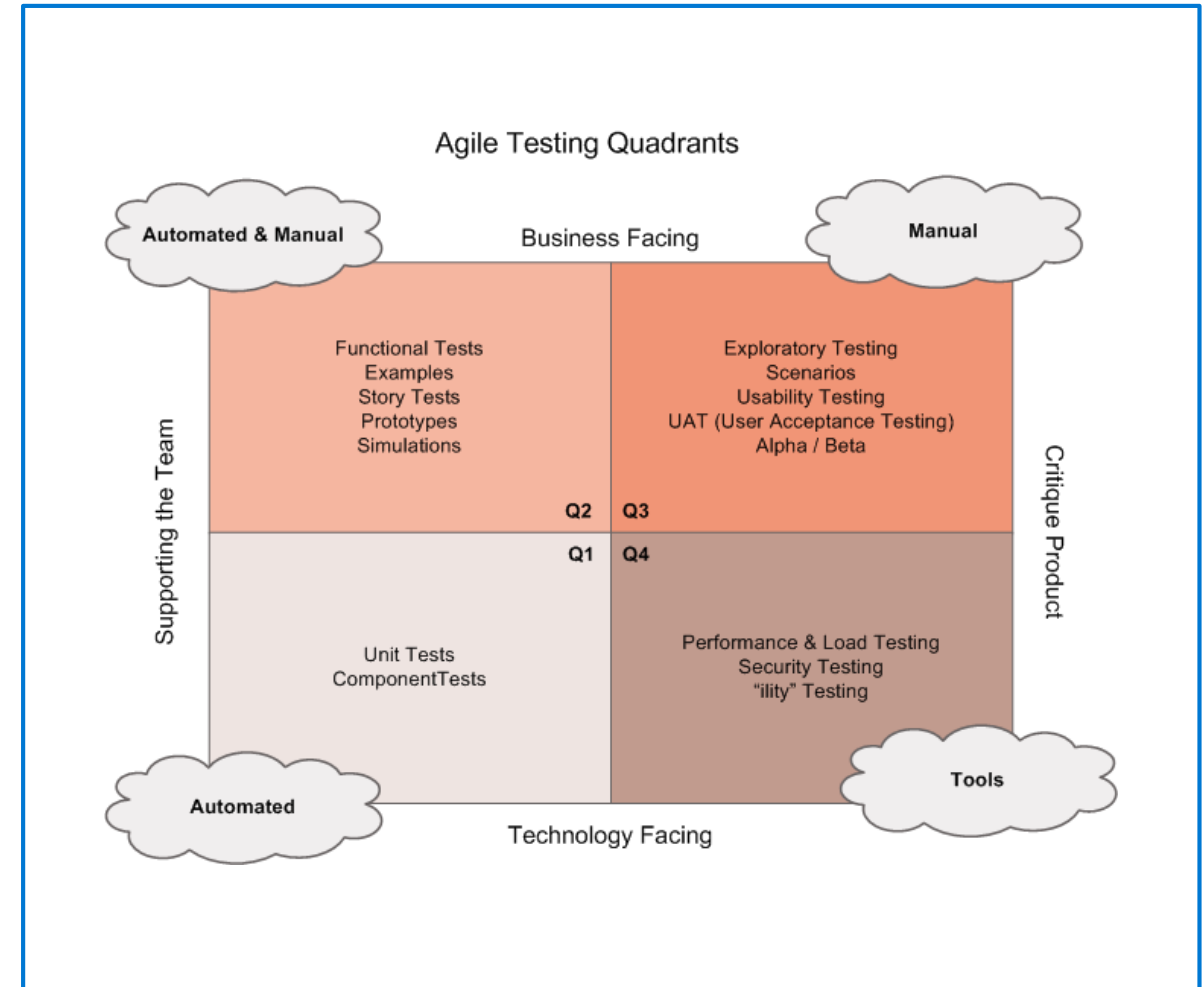


(from #mw)

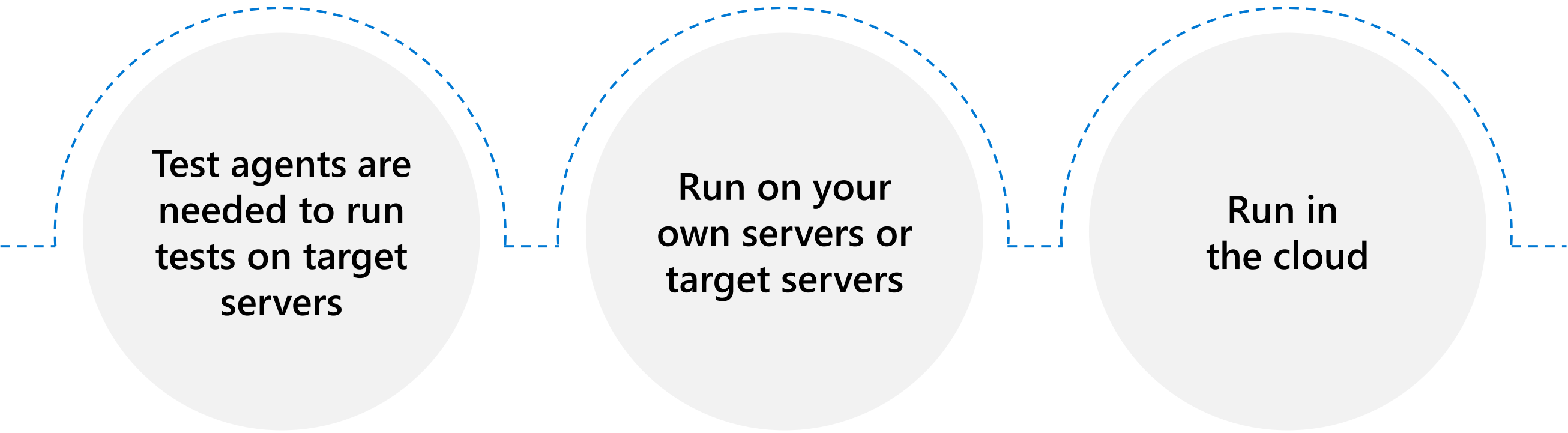
- In the *ideal* world all tests could be automated, but the world is not *ideal*
- *Increasing test coverage costs more money.*
- How can you deploy multiple times a day when some tests can't be automated?

Configure automated integration and functional test automation

- It's not possible to automate all tests
- Do not automate all your manual tests
- Rethink test strategy
- Tests should be written at the lowest level possible
- Write once, run anywhere including production system
- Product is designed for testability
- Test code is product code, only reliable tests survive
- Test ownership goes with product ownership



Setting up test infrastructure



**Test agents are
needed to run
tests on target
servers**

**Run on your
own servers or
target servers**

**Run in
the cloud**

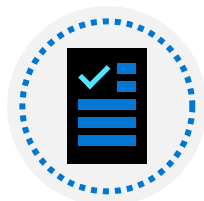
Setting up and running availability tests



Create health end points in your application



Use tools, e.g., availability tests in Application Insights, to test health endpoints

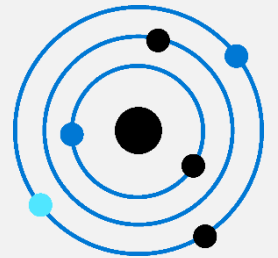


Two common types of availability tests:

URL ping test

Multi-step web test

Lesson 06: Automate inspection of health



(from #mw)

Ask, how do we keep informed about health of Build pipelines and release pipelines?

Often the people who need to be informed don't have access to Azure Dev Ops.

Automate inspection of health

Stay informed about your process and releases:



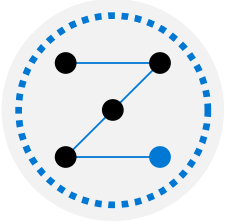
Release gates

Events,
subscriptions,
and notifications

Service hooks

Reporting

Events, subscriptions, and notifications



Actions in Azure DevOps trigger events

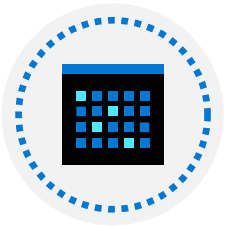
(#mw see [event types](#))



Users can subscribe to events and get notified: (#mw see project settings->notifications->new subscription)

Calling a SOAP URL

Receiving an email



Notifications can be managed centrally and personally

(#mw see user settings->notifications)

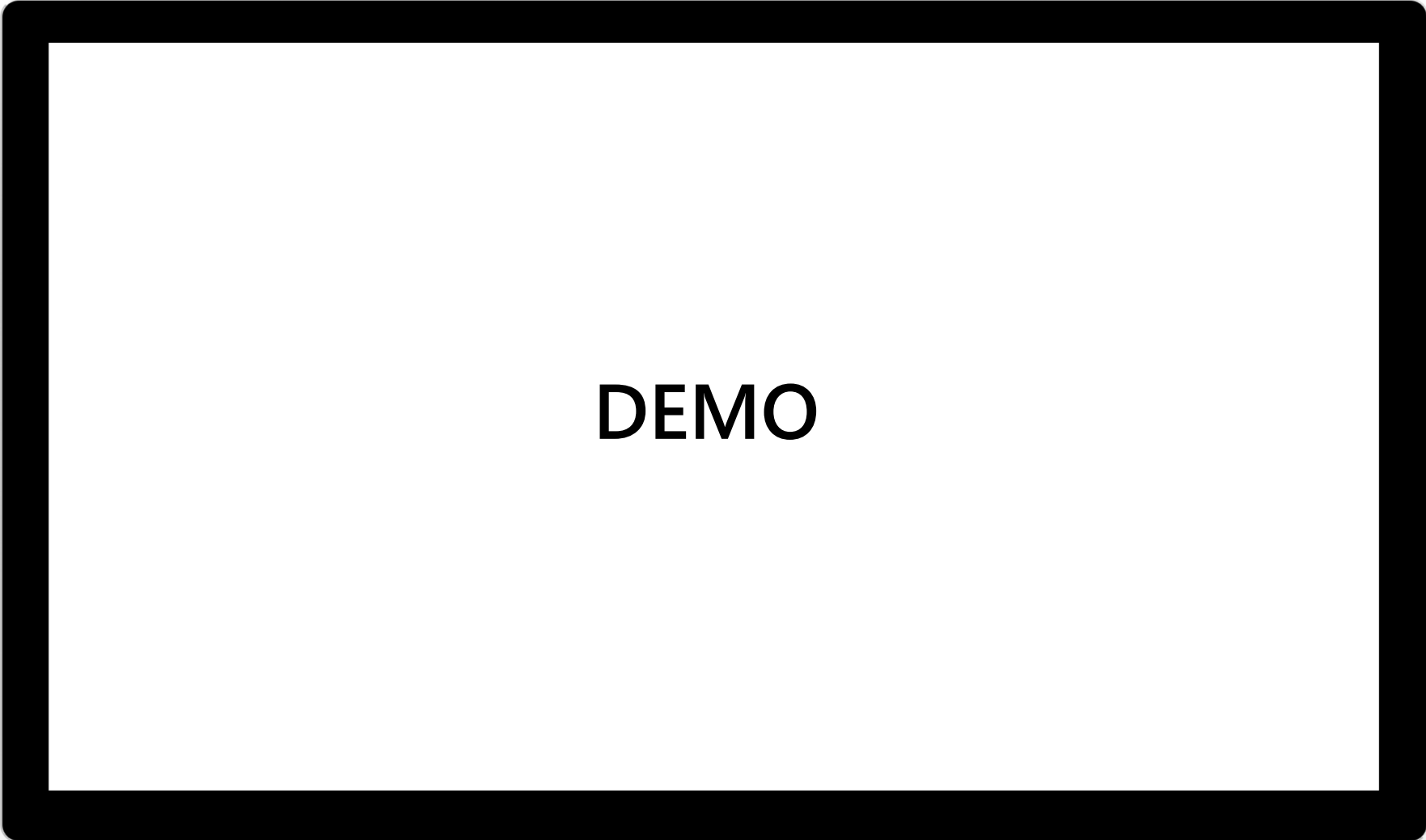
Service hooks

Service hooks enable you to perform tasks on other services when events happen

Out of the box integrations

Build and release	Collaborate	Customer support	Plan and track	Integrate
AppVeyor	Campfire	UserVoice	Trello	Azure Service Bus
Bamboo	Flowdock	Zendesk		Azure Storage
Jenkins	HipChat			Web Hooks
MyGet	Hubot			Zapier
Slack				

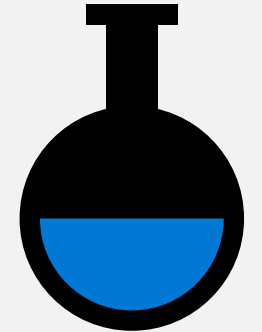
Demonstration: Setting up service hooks to monitor the pipeline



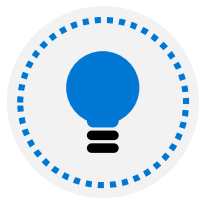
DEMO

Lesson 07: Labs

We're skipping these.
Can do outside class time



Configuring pipelines as code with YAML

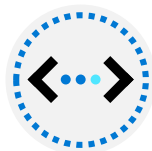


Lab overview:

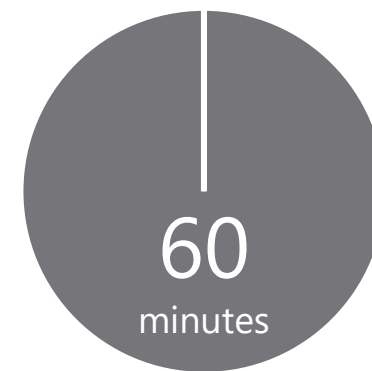
Azure DevOps also provides default templates for popular project types, as well as a YAML designer to simplify the process of defining build and release tasks.

Objectives:

Configure CI/CD pipelines as code with YAML in Azure DevOps



Duration:



Setting up and running functional tests

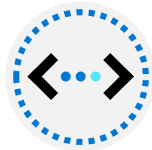


Lab overview:

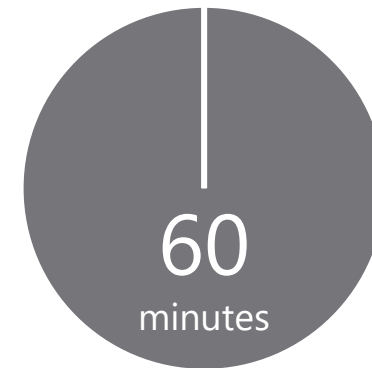
In this lab, you will learn how to execute Selenium test cases on a C# web application, as part of the Azure DevOps Release pipeline.

Objectives:

- Configure a self-hosted Azure DevOps agent
- Configure release pipeline
- Trigger build and release
- Run tests in Chrome and Firefox



Duration:



Lesson 08: Module review and takeaways



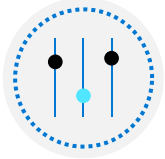
What did you learn?



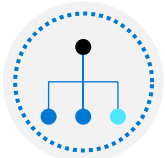
Explain the terminology used in Azure DevOps and other release management tooling



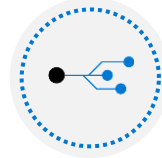
Describe what a build and release task is, what it can do, and some available deployment tasks



Explain why you sometimes need multiple release jobs in one release pipeline



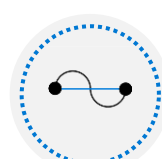
Differentiate between multi-agent and multi-configuration release job



Use release variables and stage variables in your release pipeline



Deploy to environment securely using a service connection



List the different ways to inspect the health of your pipeline and release by using, alerts, service hooks, and reports

Module review questions

1

How many deployment jobs can be run concurrently by a single agent?

2

What should you create to store values that you want to make available across multiple build and release pipelines?

3

How can you provision the agents for deployment groups in each of your VMs?

4

How can you identify a default release variable?