

Microsoft
tech·days

Kistamässan Stockholm
24-25 oktober 2018

Applied DevOps for modern apps with containers and cloud

Microsoft
tech·days

Kistamässan Stockholm
24-25 oktober 2018

Part 2: From idea to release

Agenda

- Intro to Azure DevOps
- Azure Pipelines
- Release Management

Intro to Azure DevOps

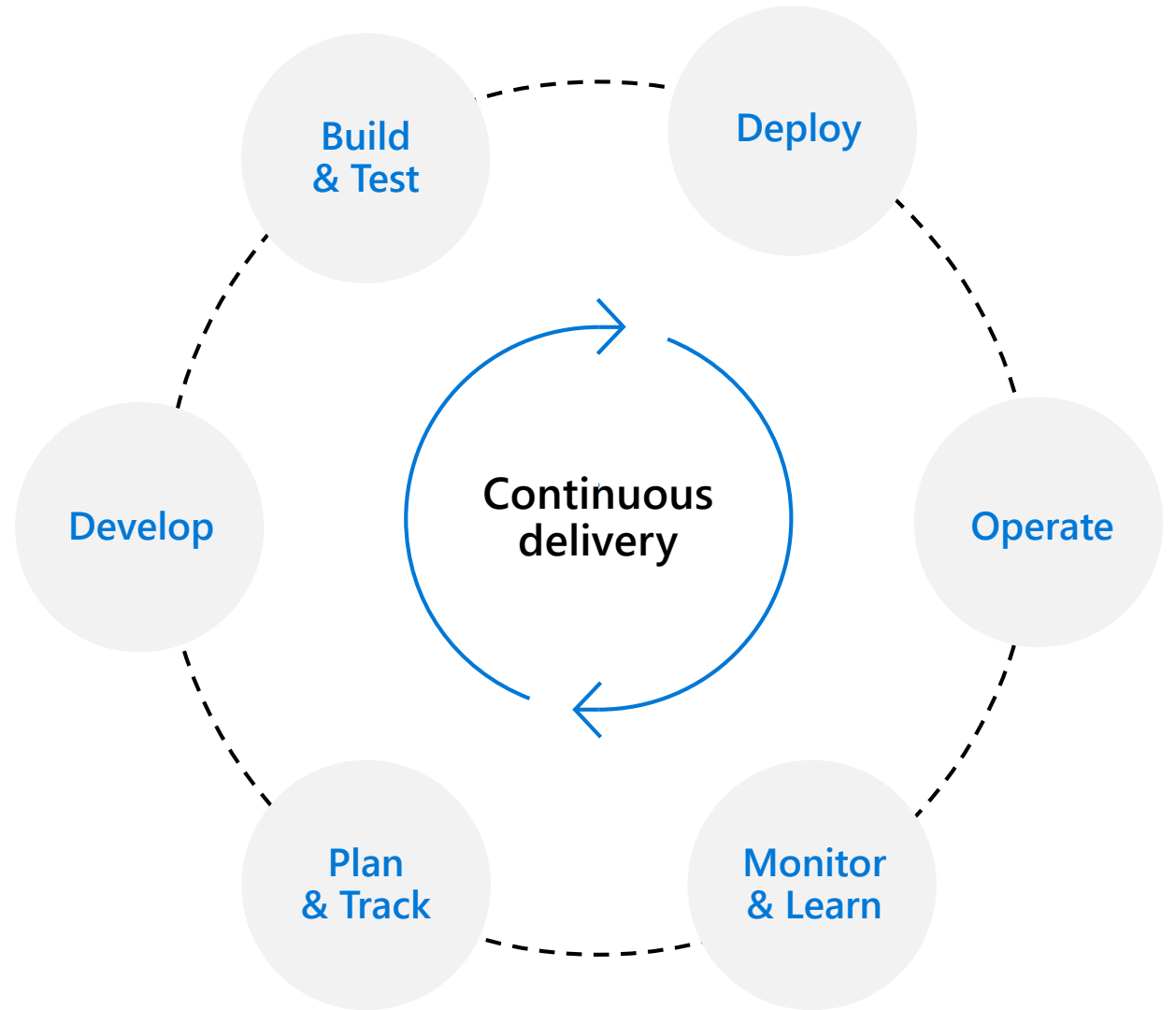
- What is DevOps?
- What is Azure DevOps?

What is DevOps?

People. Process. Products.

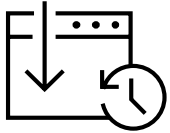
//

DevOps is the union of **people**, **process**, and **technology** to enable continuous delivery of value to your end users. //



What do I need to implement DevOps?

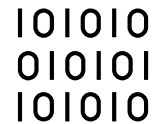
DevOps brings together people, processes, and tools, automating software delivery to provide continuous value to your users. Using Azure DevOps, you can deliver software faster and more reliably - no matter how big your IT department or what tools you're using.



Continuous integration (CI)

Improve software development quality and speed.

When you use Azure Pipelines or Jenkins to build apps in the cloud and deploy to Azure, each time you commit code, it's automatically built and tested and bugs are detected faster.



Continuous delivery (CD)

Ensures that code and infrastructure are always in a production-deployable state.

By combining continuous integration and infrastructure as code (IaC), you'll achieve identical deployments and the confidence to deploy to production at any time.



Continuous deployment with CI/CD

With continuous deployment, you can automate the entire process from code commit to production if your CI/CD tests are successful.

Using CI/CD practices, paired with monitoring tools, you'll be able to safely deliver features to your customers as soon as they're ready.

Azure DevOps



Azure Boards

Plan, track, and discuss work across teams, deliver value to your users faster.



Azure Repos

Unlimited cloud-hosted private Git repos. Collaborative pull requests, advanced file management, and more.



Azure Pipelines

CI/CD that works with any language, platform, and cloud. Connect to GitHub or any Git provider and deploy continuously to any cloud.



Azure Test Plans

The test management and exploratory testing toolkit that lets you ship with confidence.



Azure Artifacts

Create, host, and share packages. Easily add artifacts to CI/CD pipelines.

Azure Boards

Track work with Kanban boards, backlogs, team dashboards, and custom reporting



Connected from idea to release

Track all your ideas at every development stage and keep your team aligned with all code changes linked directly to work items.



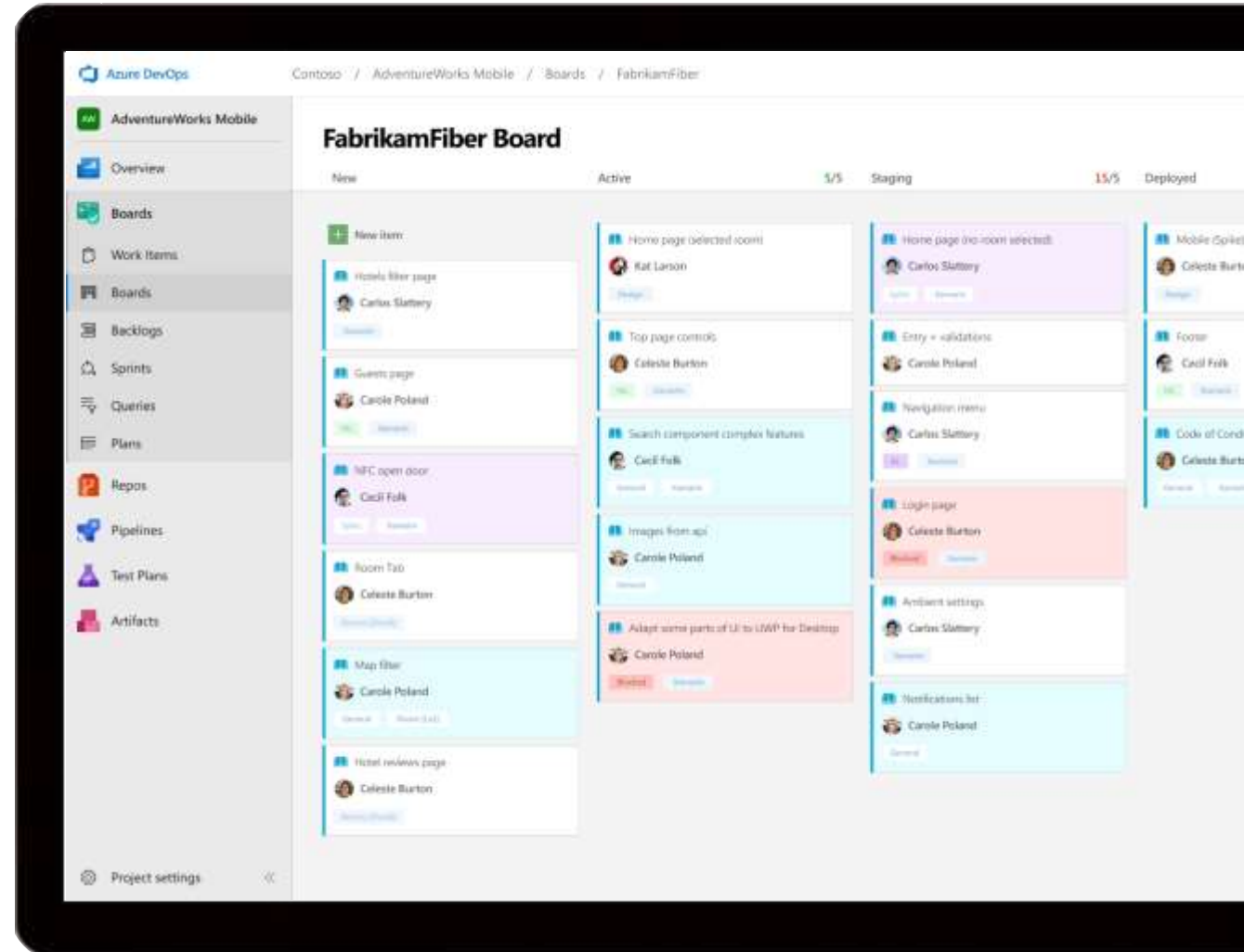
Scrum ready

Use built-in scrum boards and planning tools to help your teams run sprints, stand-ups, and planning meetings.



Project insights

Gain new insights into the health and status of your project with powerful analytics tools and dashboard widgets.



Azure Repos

Unlimited private Git repo hosting and support for TFVC that scales from a hobby project to the world's largest Git repositories



Works with your Git client

Securely connect with and push code into your Git repos from any IDE, editor, or Git client.



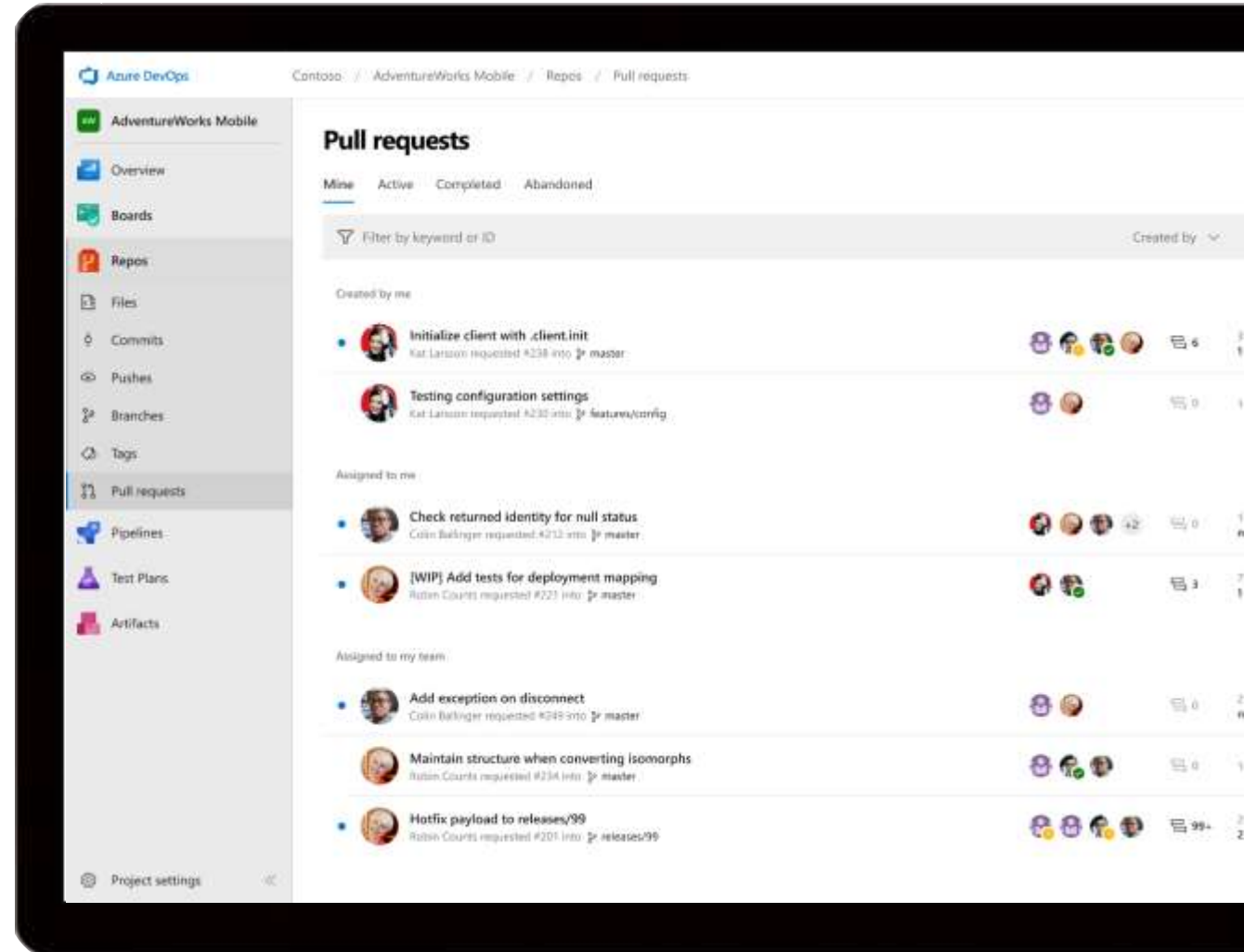
Web hooks and API integration

Add validations and extensions from the marketplace or build your own using web hooks and REST APIs.



Semantic code search

Quickly find what you're looking for with code-aware search that understands classes and variables.



Azure Pipelines

Cloud-hosted pipelines for Linux, Windows and macOS, with unlimited minutes for open source



Any language, any platform, any cloud

Build, test, and deploy Node.js, Python, Java, PHP, Ruby, C/C++, .NET, Android, and iOS apps. Run in parallel on Linux, macOS, and Windows. Deploy to Azure, AWS, GCP or on-premises



Extensible

Explore and implement a wide range of community-built build, test, and deployment tasks, along with hundreds of extensions from Slack to SonarCloud. Support for YAML, reporting and more



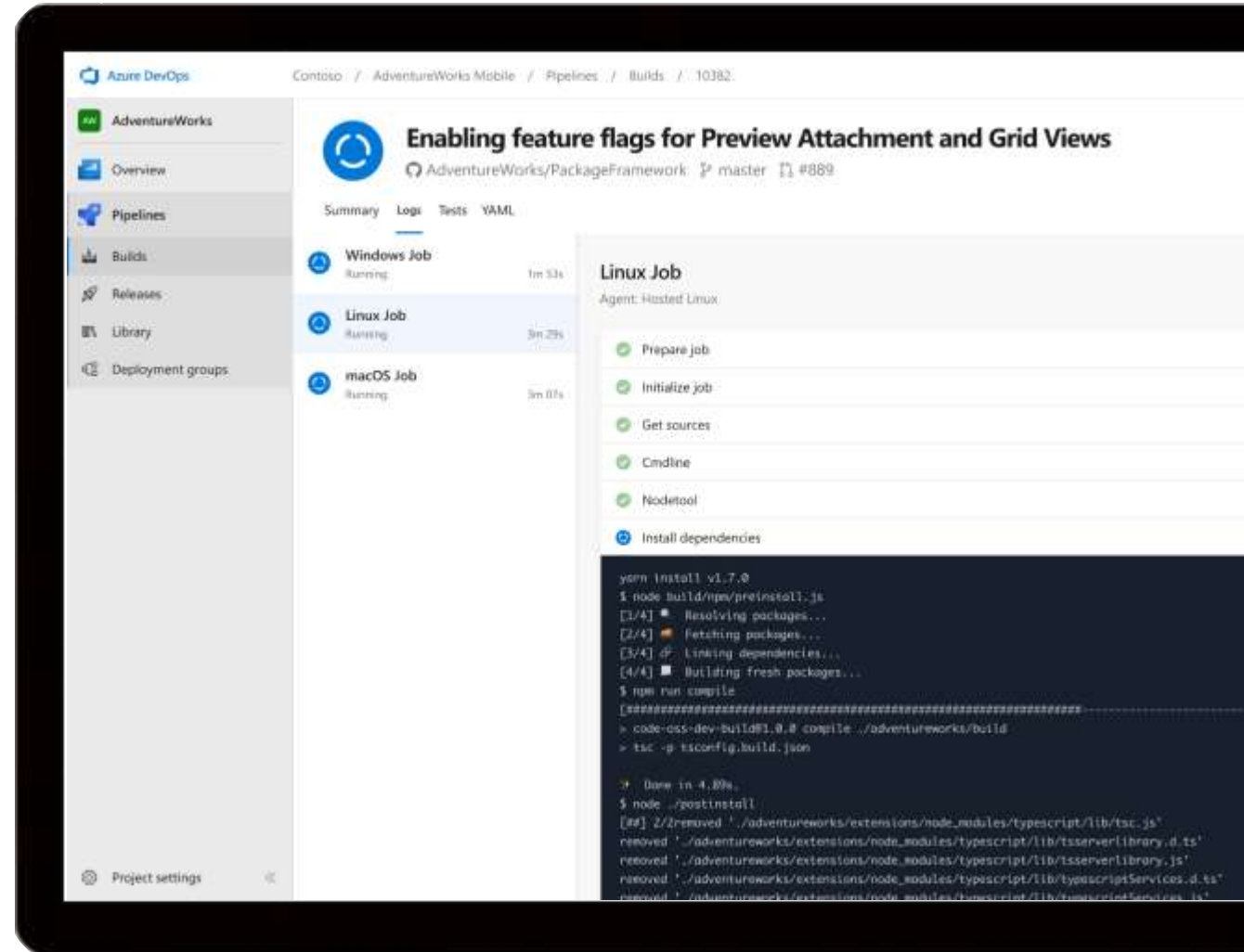
Containers and Kubernetes

Easily build and push images to container registries like Docker Hub and Azure Container Registry. Deploy containers to individual hosts or Kubernetes.



Best-in-class for open source

Ensure fast continuous integration/continuous delivery (CI/CD) pipelines for every open source project. Get unlimited build minutes for all open source projects with up to 10 free parallel jobs across Linux, macOS and Windows



Azure Test Plans

Get end-to-end traceability. Run tests and log defects from your browser. Track and assess quality throughout your testing lifecycle.



Capture rich data

Capture rich scenario data as you execute tests to make discovered defects actionable. Explore user stories without test cases or test steps. You can create test cases directly from your exploratory test sessions.



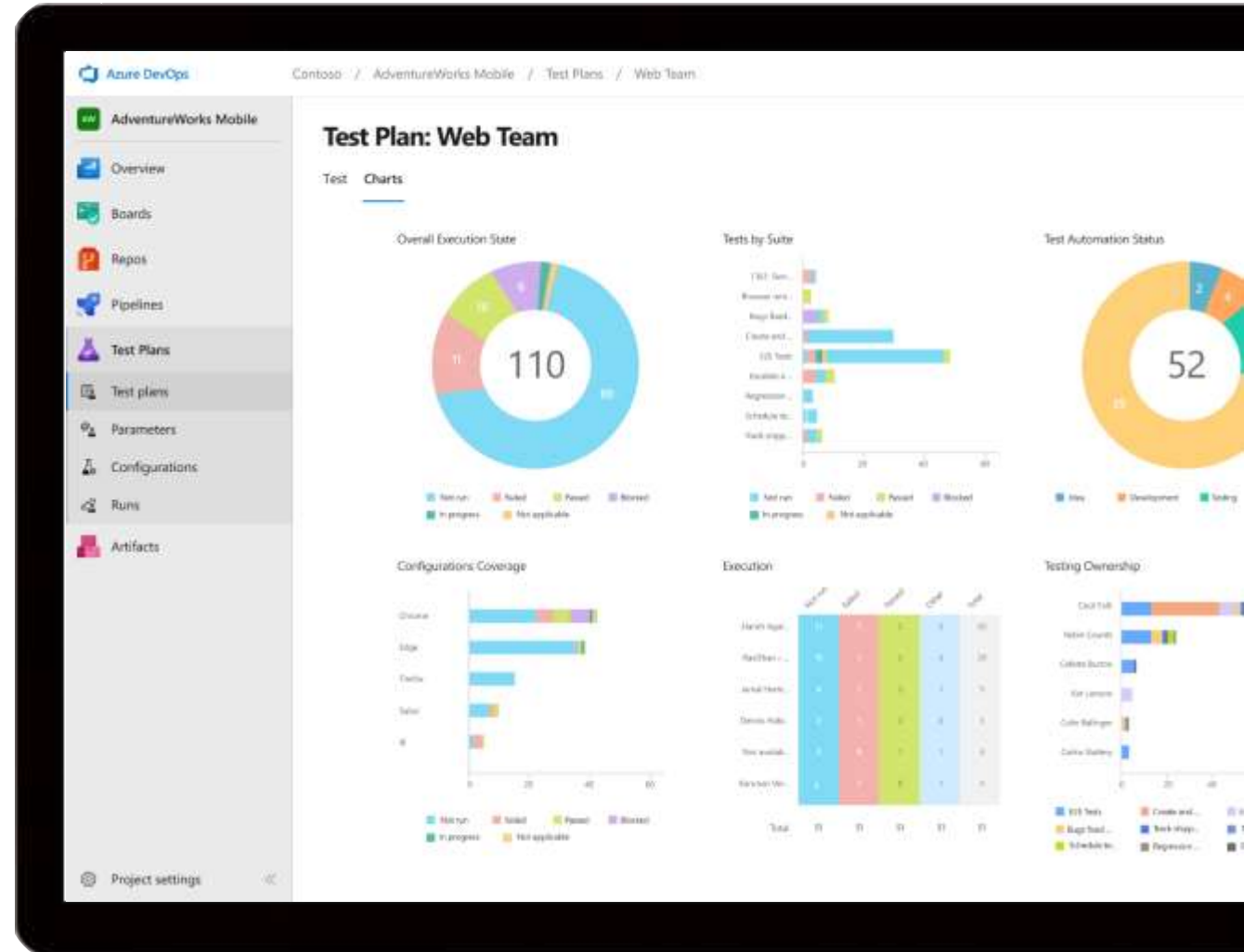
Test across web and desktop

Test your application where it lives. Complete scripted tests across desktop or web scenarios. Test on-premises application from the cloud and vice-versa.



Get end-to-end traceability

Leverage the same test tools across your engineers and user acceptance testing stakeholders. Pay for the tools only when you need them.



Azure Artifacts

Create and share Maven, npm, and NuGet package feeds from public and private sources – fully integrated into CI/CD pipelines



Manage all package types

Get universal artifact management for Maven, npm, and NuGet.



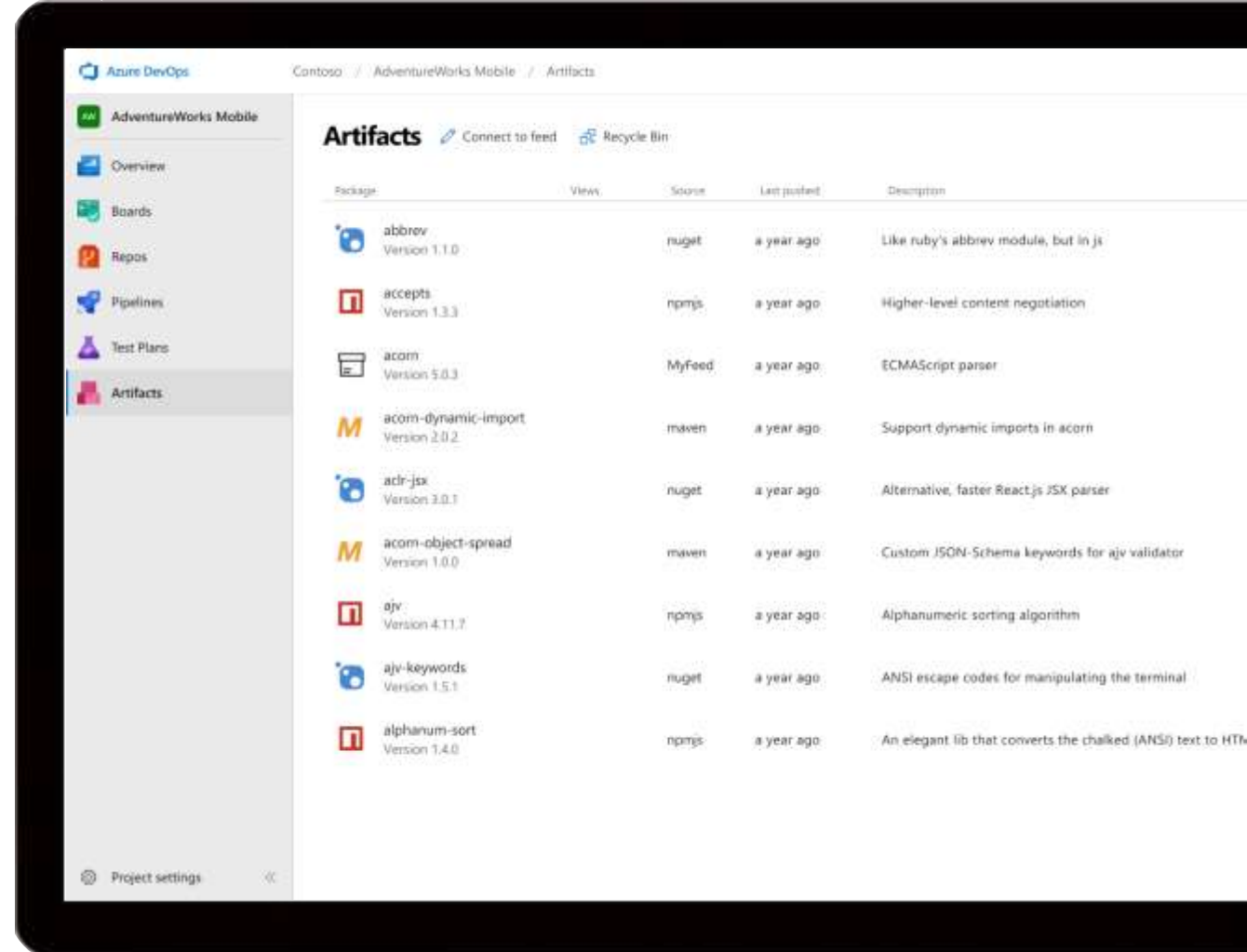
Add packages to any pipeline

Share packages, and use built-in CI/CD, versioning, and testing.



Share code efficiently

Easily share code across small teams and large enterprises.



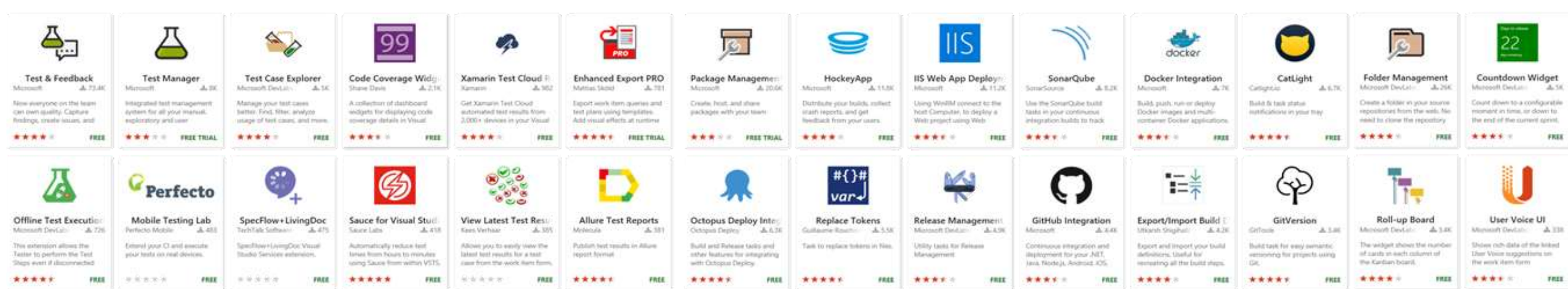
Visual Studio Marketplace

Browse, discover and install

The **Visual Studio Marketplace** is a new destination and the exclusive place for purchasing subscriptions and for discovering extensions for Azure DevOps and Team Foundation Server.

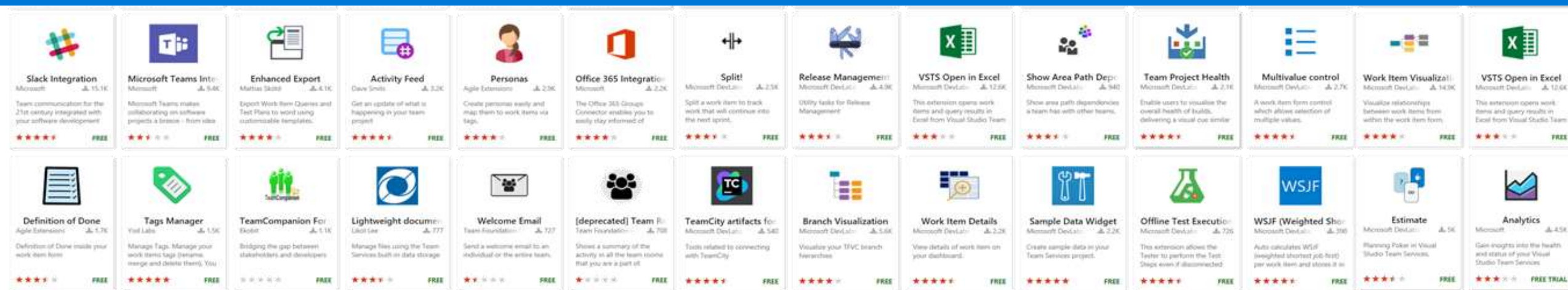
You can find extensions from within the product or on the web and you can install them with a few clicks.



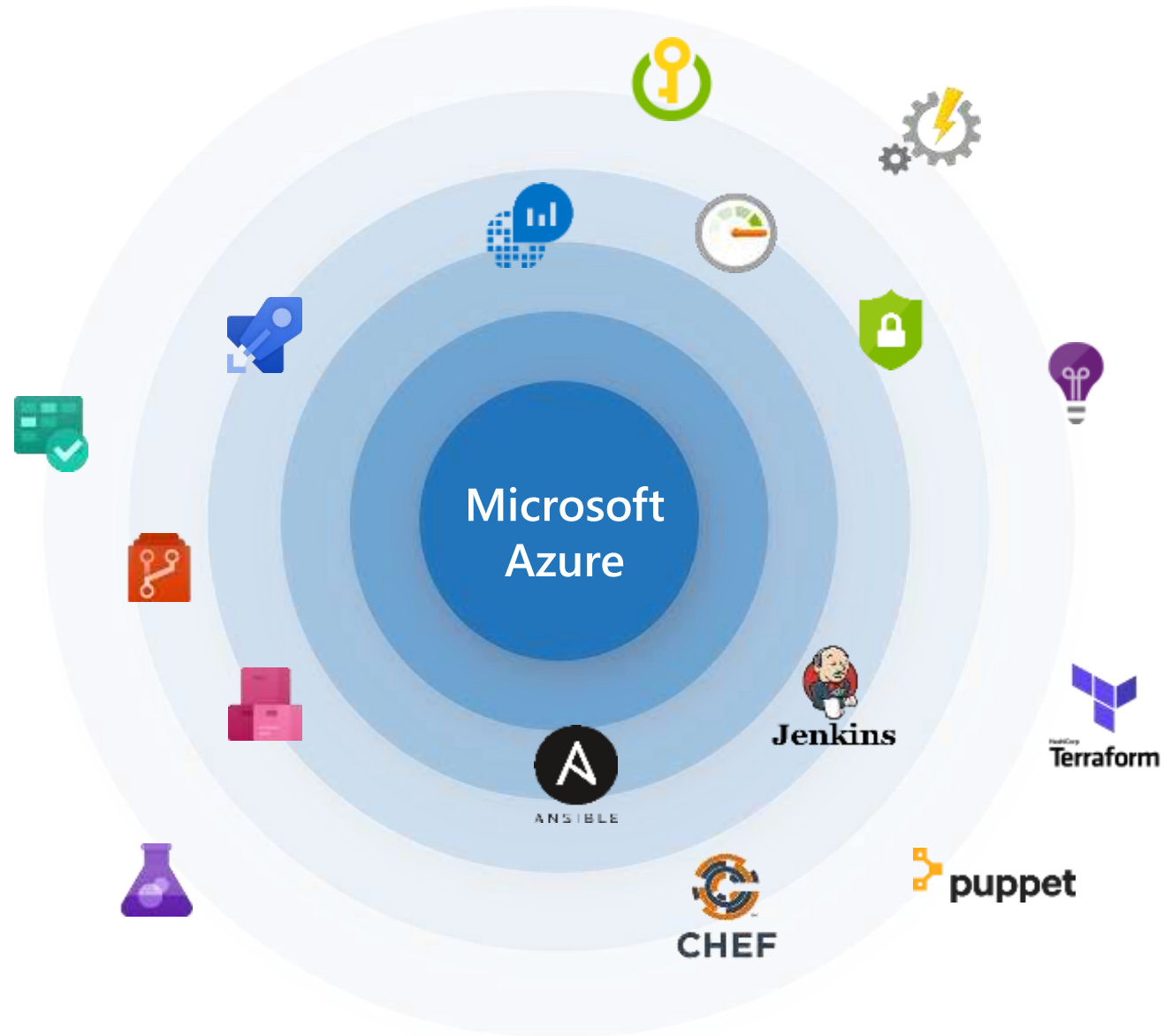


Extensions

Over 500 extensions for Azure DevOps and TFS covering all elements of the project lifecycle and integrations with popular commercial and open source tooling



DevOps in the Azure Ecosystem



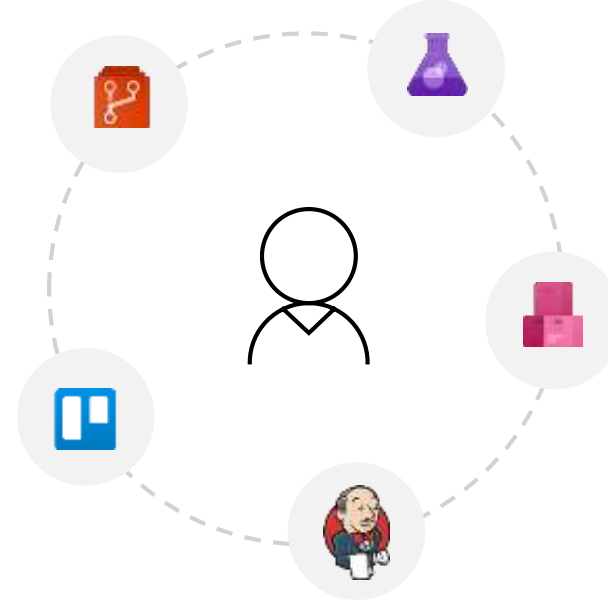
Azure DevOps: Choose what you love

Your tools, languages, and clouds

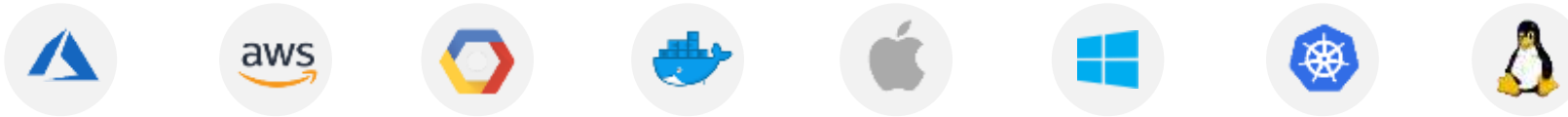
Azure DevOps lets developers choose the tools and languages that are right for them



Mix and match to create workflows with tools from Microsoft, open source or your favorite 3rd party tools



Target any cloud, on-prem or both and deploy to the servers you need



Azure Pipelines

- Azure Pipelines
- Continuous Integration
- Continuous Delivery

Azure Pipelines

Cloud-hosted pipelines for Linux, Windows and macOS, with unlimited minutes for open source



Any language, any platform, any cloud

Build, test, and deploy Node.js, Python, Java, PHP, Ruby, C/C++, .NET, Android, and iOS apps. Run in parallel on Linux, macOS, and Windows. Deploy to Azure, AWS, GCP or on-premises



Extensible

Explore and implement a wide range of community-built build, test, and deployment tasks, along with hundreds of extensions from Slack to SonarCloud. Support for YAML, reporting and more



Containers and Kubernetes

Easily build and push images to container registries like Docker Hub and Azure Container Registry. Deploy containers to individual hosts or Kubernetes.

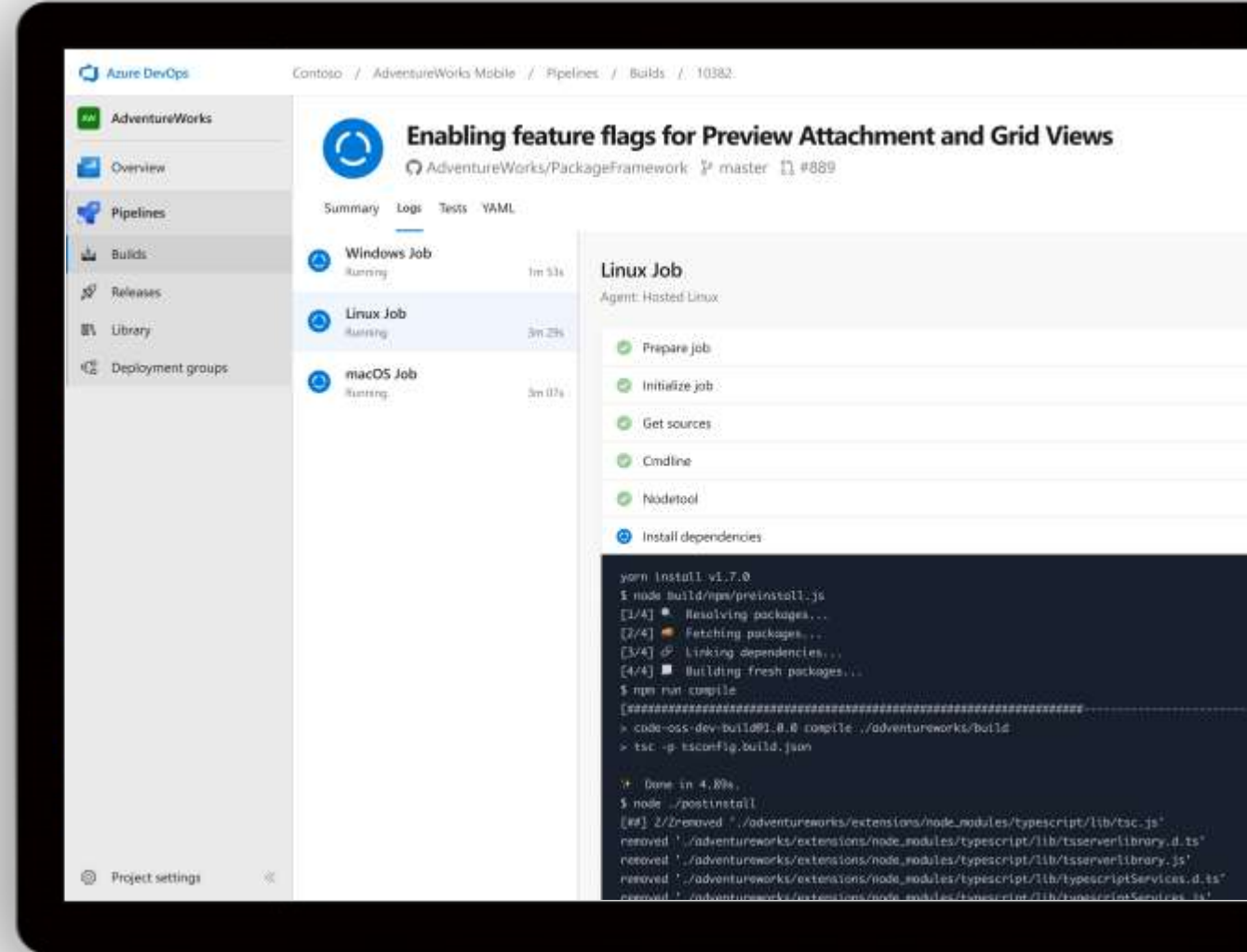


Best-in-class for open source

Ensure fast continuous integration/continuous delivery (CI/CD) pipelines for every open source project. Get unlimited build minutes for all open source projects with up to 10 free parallel jobs across Linux, macOS and Windows



<https://azure.com/pipelines>



Azure Pipelines as Code

With YAML you can define your build process as code and have it close to the application



Model pipeline in code

Pipelines are made of one or more jobs and may include resources and variables. Jobs are made of one or more steps plus some job-specific data. Steps can be tasks, scripts, or references to external templates.



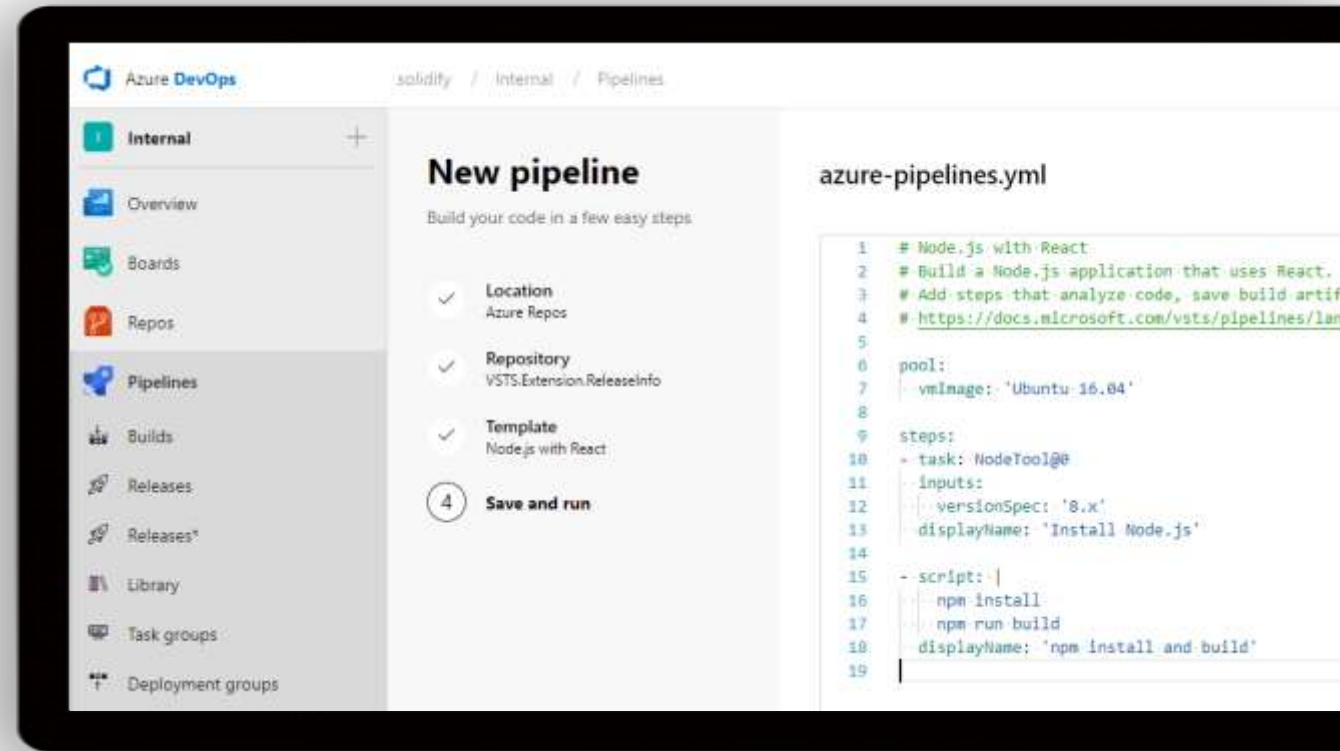
New pipeline wizard

A new wizard simplifies this process of creating YAML-based build pipelines with GitHub and Azure Repos. Once you choose a repository to build, a pipeline will automatically be created if it contains a YAML file. Otherwise, Azure Pipelines will analyze your repository and recommend a YAML-based template for building your project. Note: currently a preview feature



Status

Currently YAML is available for builds in Azure DevOps services. Support for releases is planned for preview in Q3 2018 (Azure DevOps Services). On-prem support is coming in Azure DevOps Server.



 <https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema>

Azure Pipelines – any tool, any platform

Flexible agent model enables running any tool on any platform



Generic task runner

The core of the Azure Pipeline execution is the jobs engine that can execute any task on the given platform. Commands are (generally) run locally on the agent, giving great control over the processing. The tasks can be written to work on any platform and the marketplace contains over 500 task extensions



Cross-platform agents

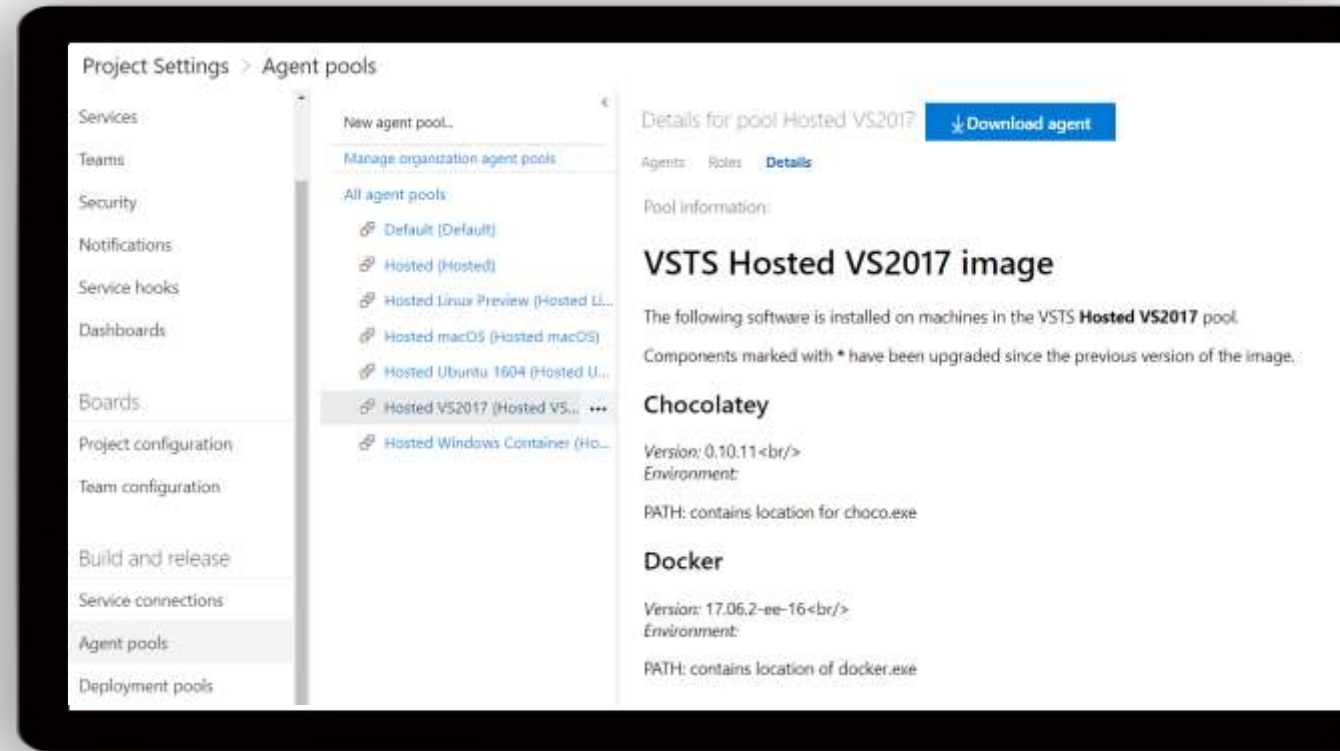
To truly allow us to run any tool on any platform there are plenty of options for running tasks with cross platform agents.

Azure Pipelines agents are available as hosted or you can run your own self-hosted agents.



XAML build

Xaml build is still supported but requires standing up a self-hosted build server.



<https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/agents>

Azure Pipelines – governance

With great powers comes great responsibilities. Azure pipelines gives you control and traceability throughout the release process.



Control releases with approvals

When a release is created from a release pipeline that defines approvals, the deployment stops at each point where approval is required until the specified approver grants approval or rejects the release



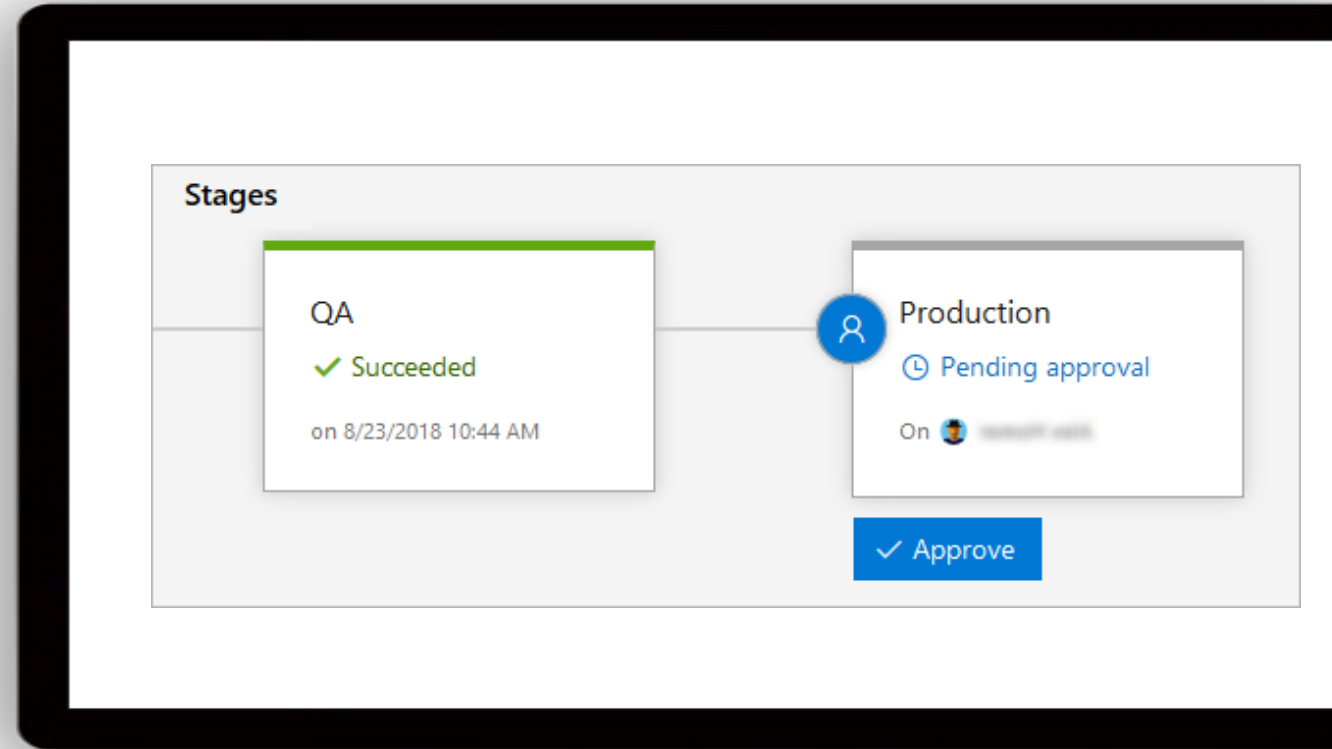
Automate approval with gates

Gates allow automatic collection of health signals from external services, and then promote the release when all the signals are successful at the same time or stop the deployment on timeout



Implement release lifecycles

Versioning of artifacts, builds and deployments help to make the release process clear. Which branches to build or artifacts to deploy can be controlled using filters. Released builds are automatically retained for traceability



<https://docs.microsoft.com/en-us/azure/devops/pipelines/release/what-is-release-management>

Azure DevOps Projects

Cloud-hosted pipelines for Linux, Windows and macOS, with unlimited minutes for open source



Create a full DevOps pipeline with 3 easy steps from the Azure Portal



Start with a Git repo and any source language



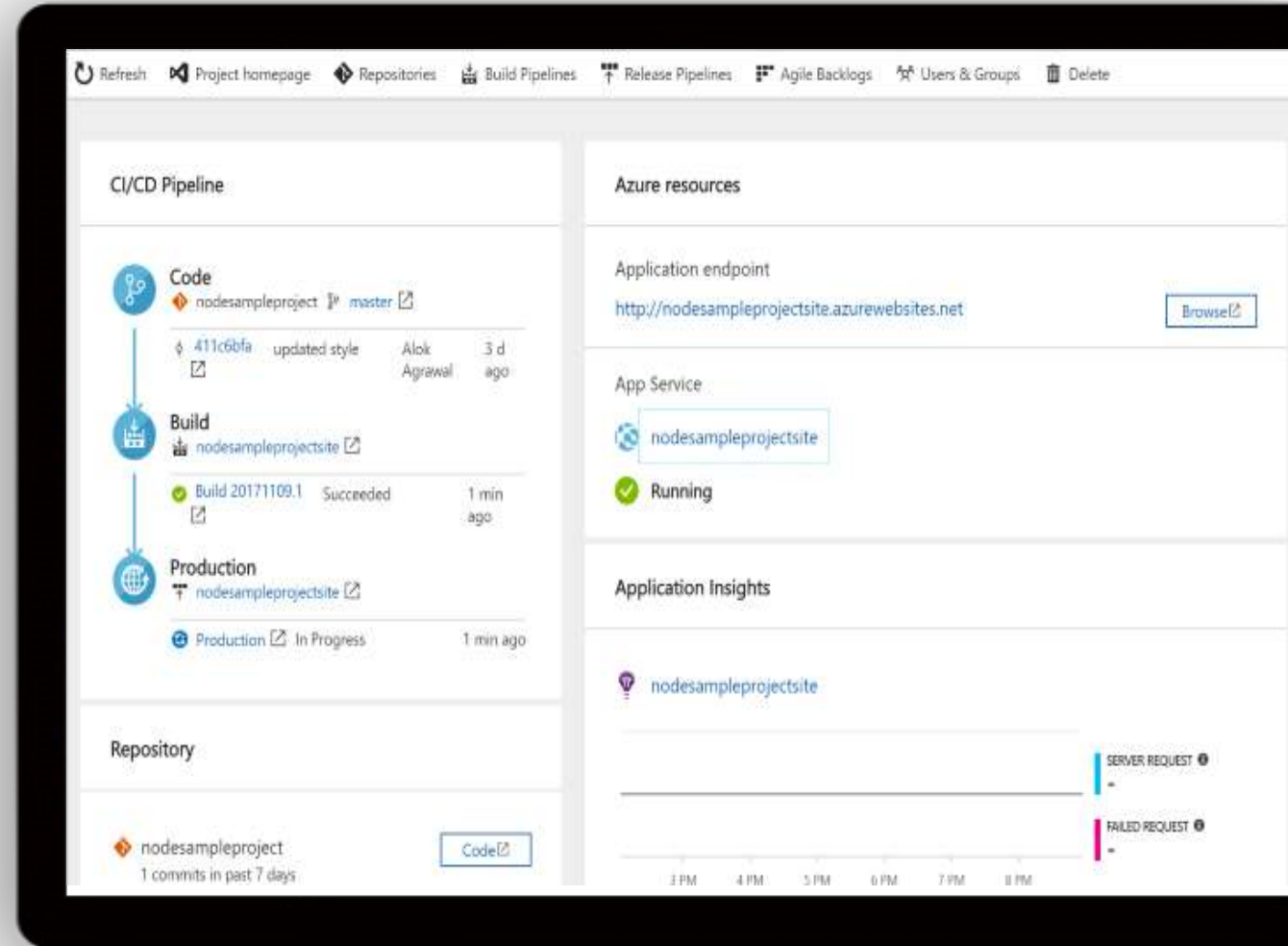
Web apps, Kubernetes, soon VMs and more.



Customize, extend and scale when needed.



<https://azure.microsoft.com/en-us/features/devops-projects/>



Continuous Integration

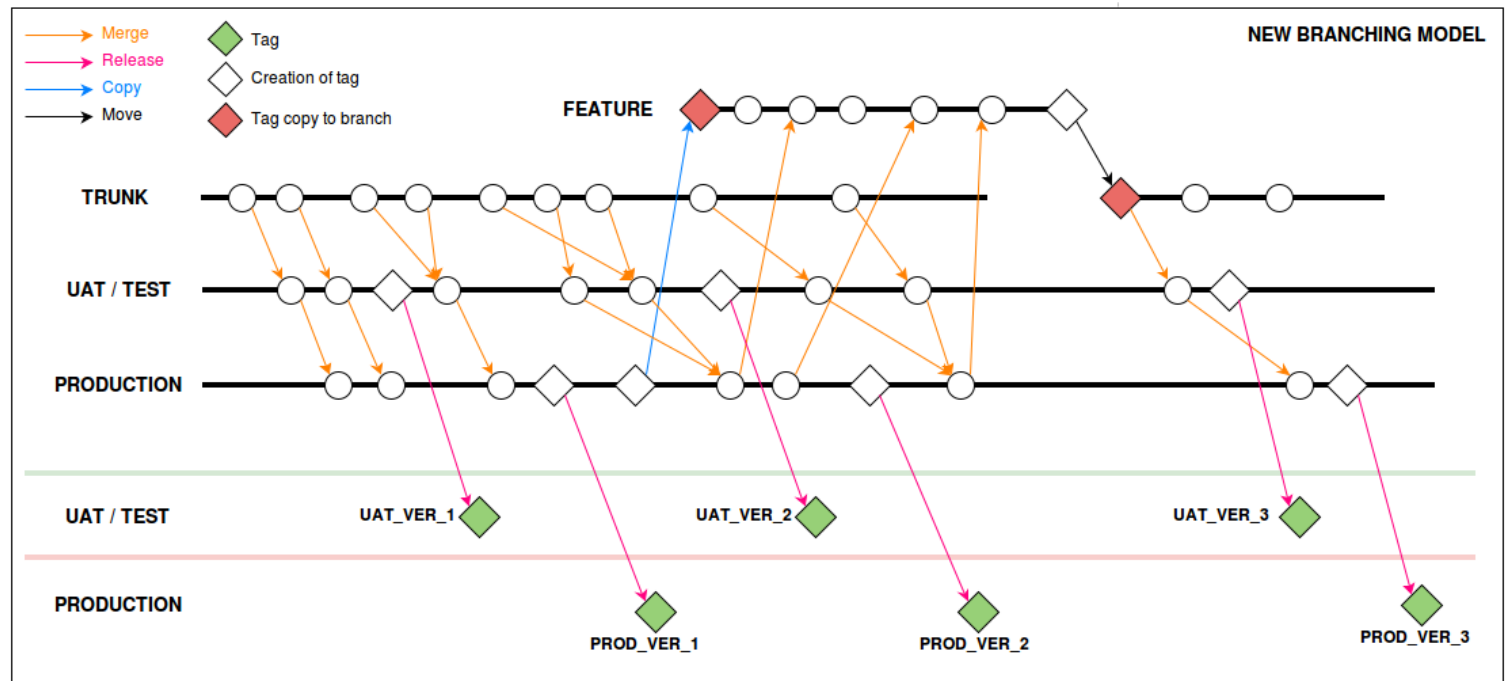
Continuous Integration

Continuous Integration (CI) is a development practice that requires developers to **integrate** code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

- Important principles
 - Version control
 - Branching strategy
 - Branch policies
 - Build automation
 - Artifacts

Branching strategy


- Use what makes sense for your needs
 - Spend your creativity on your app, not your branching strategy (anonymous)
- Several good options
 - GitFlow
 - Feature/topic branches
 - Trunk based development



Branch policies

- Protect master (or other branches)
- Common requirements
 - Require code review
 - Require work item
 - CI build must pass

Policies for: MyHealthClinic > delete-me > master

 Save changes  Discard changes

Protect this branch

- Setting a Required policy will enforce the use of pull requests when updating the branch
- Setting a Required policy will prevent branch deletion
- Manage permissions for this branch on the [Security page](#)

☒ Require a minimum number of reviewers

Require approval from a specified number of reviewers on pull requests.

Minimum number of reviewers

☐ Allow users to approve their own changes.

☐ Allow completion even if some reviewers vote "Waiting" or "Reject".

☐ Reset code reviewer votes when there are new changes.

☒ Check for linked work items

Encourage traceability by checking for linked work items on pull requests.

Policy requirement

☒ Required

Block pull requests from being completed unless they have at least one linked work item.

☐ Optional

Warn if there are no linked work items, but allow pull requests to be completed.

☐ Check for comment resolution


Check to see that all comments have been resolved on pull requests.

☐ Enforce a merge strategy

Require a specific type of merge when pull requests are completed.

Build validation

Validate code by pre-merging and building pull request changes

 Add build policy

Build pipeline	Requirement	Path filter	Expiration	Trigger	
 delete-me	Required	No filter	Expires after 12 hours	Automatic	 Enabled

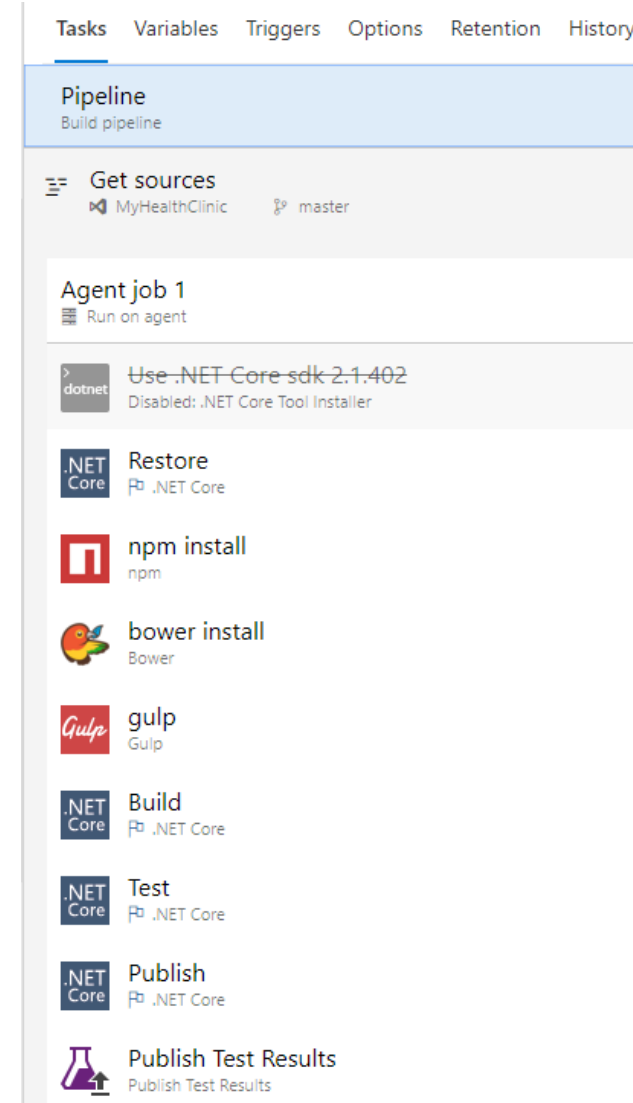
Require approval from external services

Require third party services to post successful status to complete pull requests. [Learn more](#)

 Add status policy

Build automation

- Core steps
 - Restore packages
 - Compile
 - Run unit tests (L0-1)
 - Run code analysis
 - Package artifacts
- Azure Pipelines run on hosted or private agents
 - And on Window, Mac or Linux
- Very extensible
 - But... be aware of task implementations



Demo

Continuous Integration

Continuous Delivery

Continuous Delivery

Continuous Delivery (CD) is a software development discipline where you build software in such a way that the software can be released to production at any time.

- Practices
 - Infrastructure as code
 - Automated deployment
 - Configuration
 - Feature toggles

Infrastructure as Code

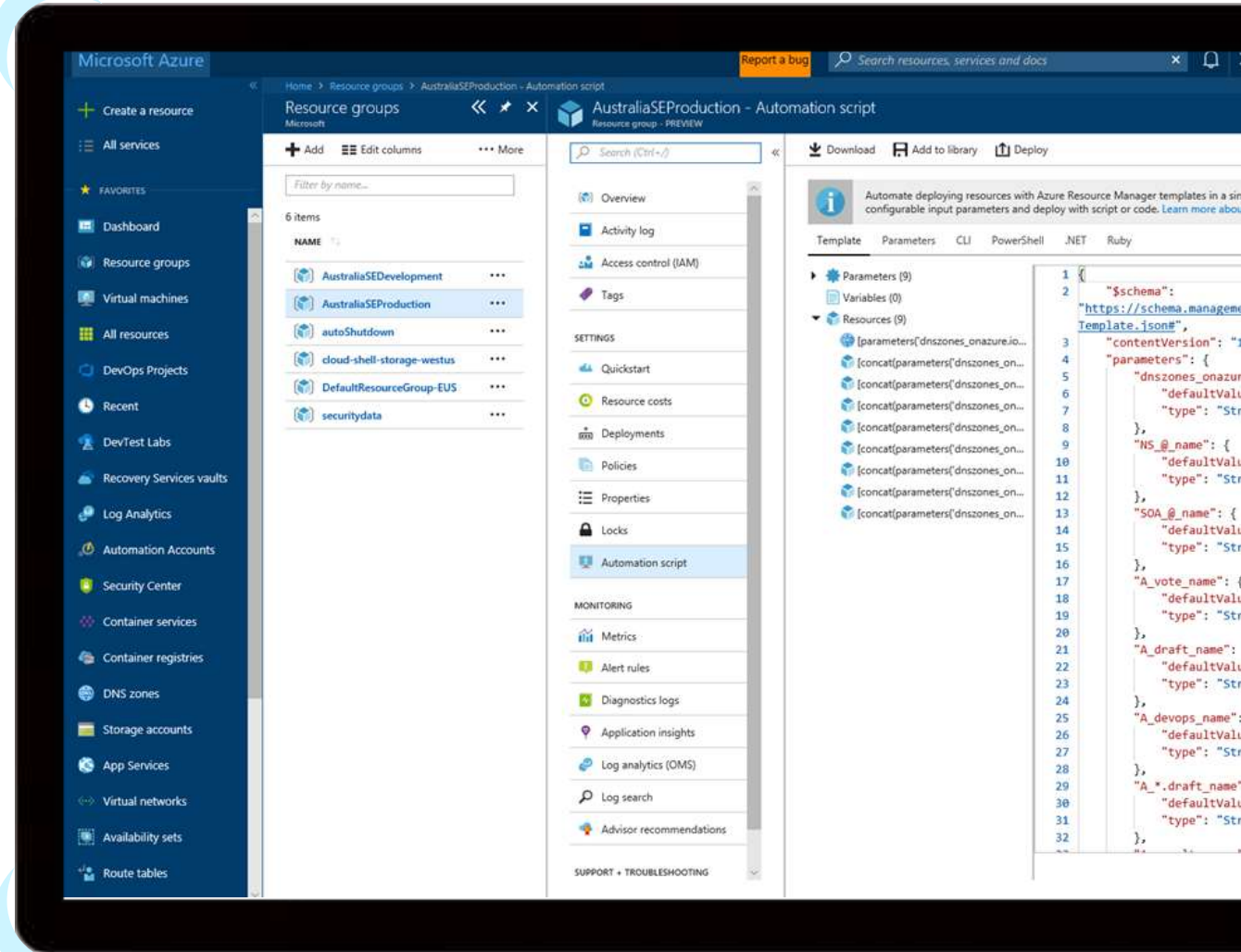
- Infrastructure to the team!
- Environments and configuration checked into version control
- Benefits of IOC
 - Repeatable
 - Reduce risk
 - Traceable
 - Elastic

An example: Azure Resource Manager

Infrastructure as Code, built-in with Azure Resource Manager

Use Azure Automation & Config to automate repetitive tasks

Support for DevOps tool integrations and OSS tooling such as Terraform, Ansible & Chef



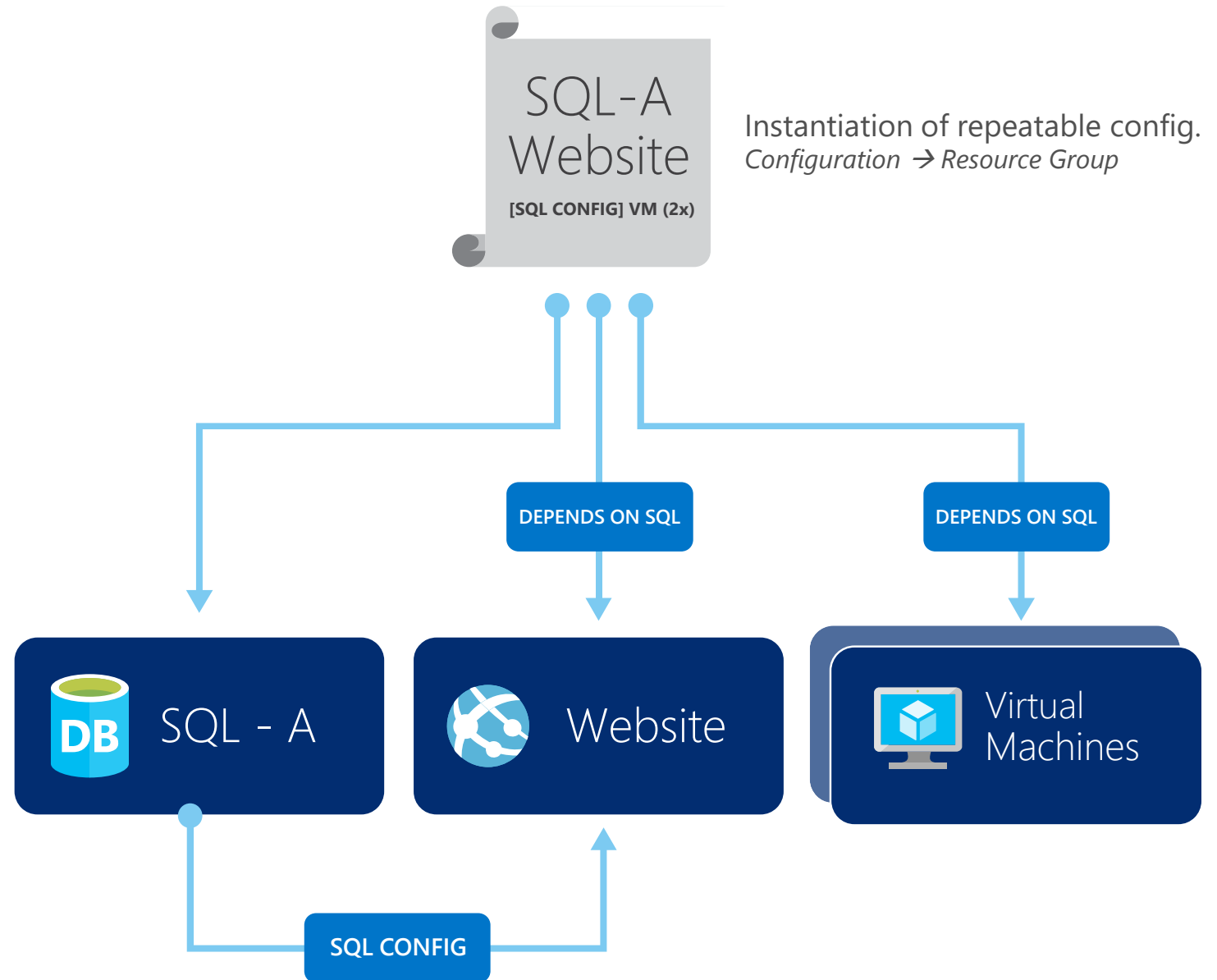
ARM Templates

ARM Templates can:

- Ensure Idempotency
- Simplify Orchestration
- Simplify Roll-back
- Provide Cross-Resource Configuration and Update Support

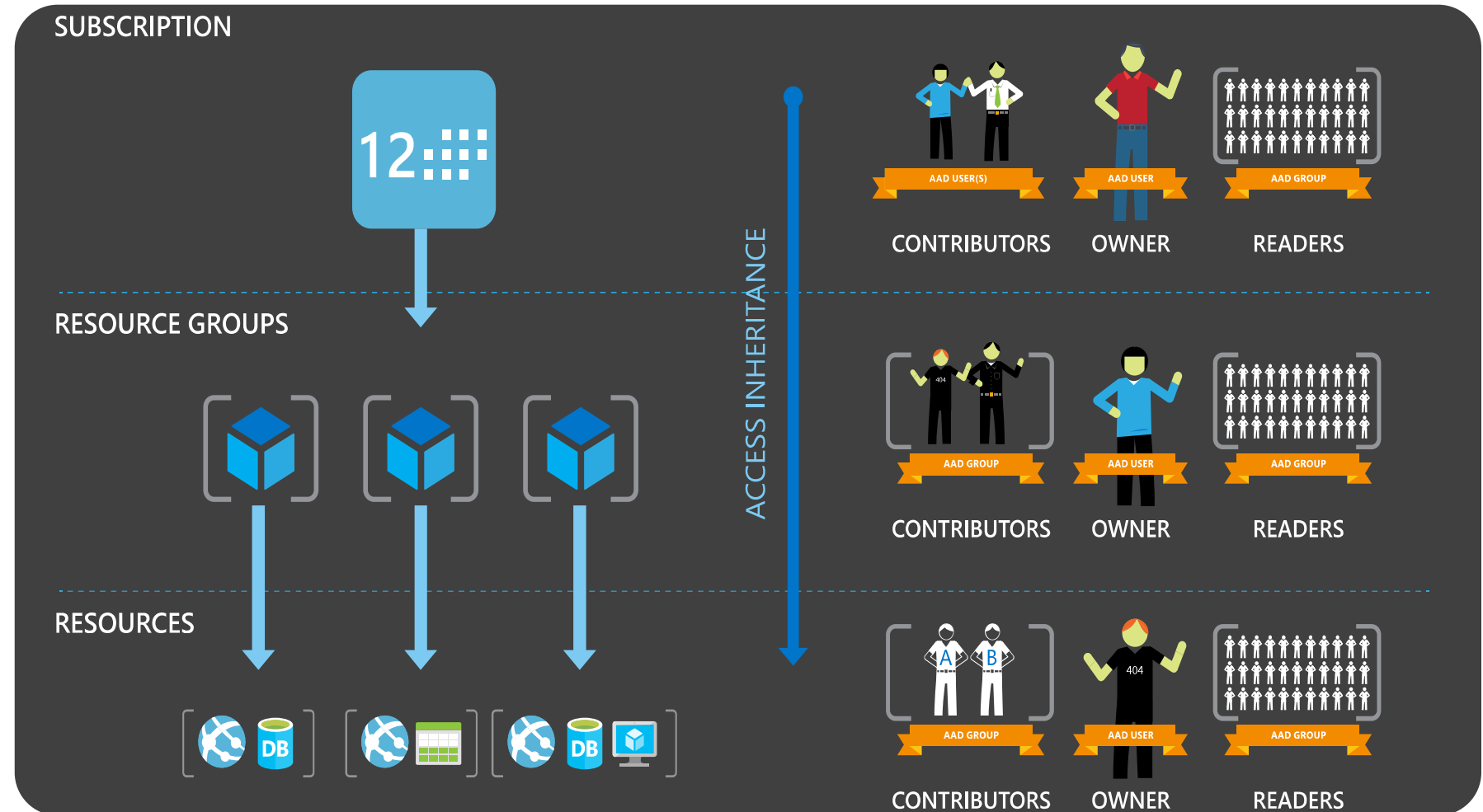
ARM Templates are:

- Source file, checked-in
- Specifies resources and dependencies (VMs, WebSites, DBs) and connections (config, LB sets)
- Parametized input/output

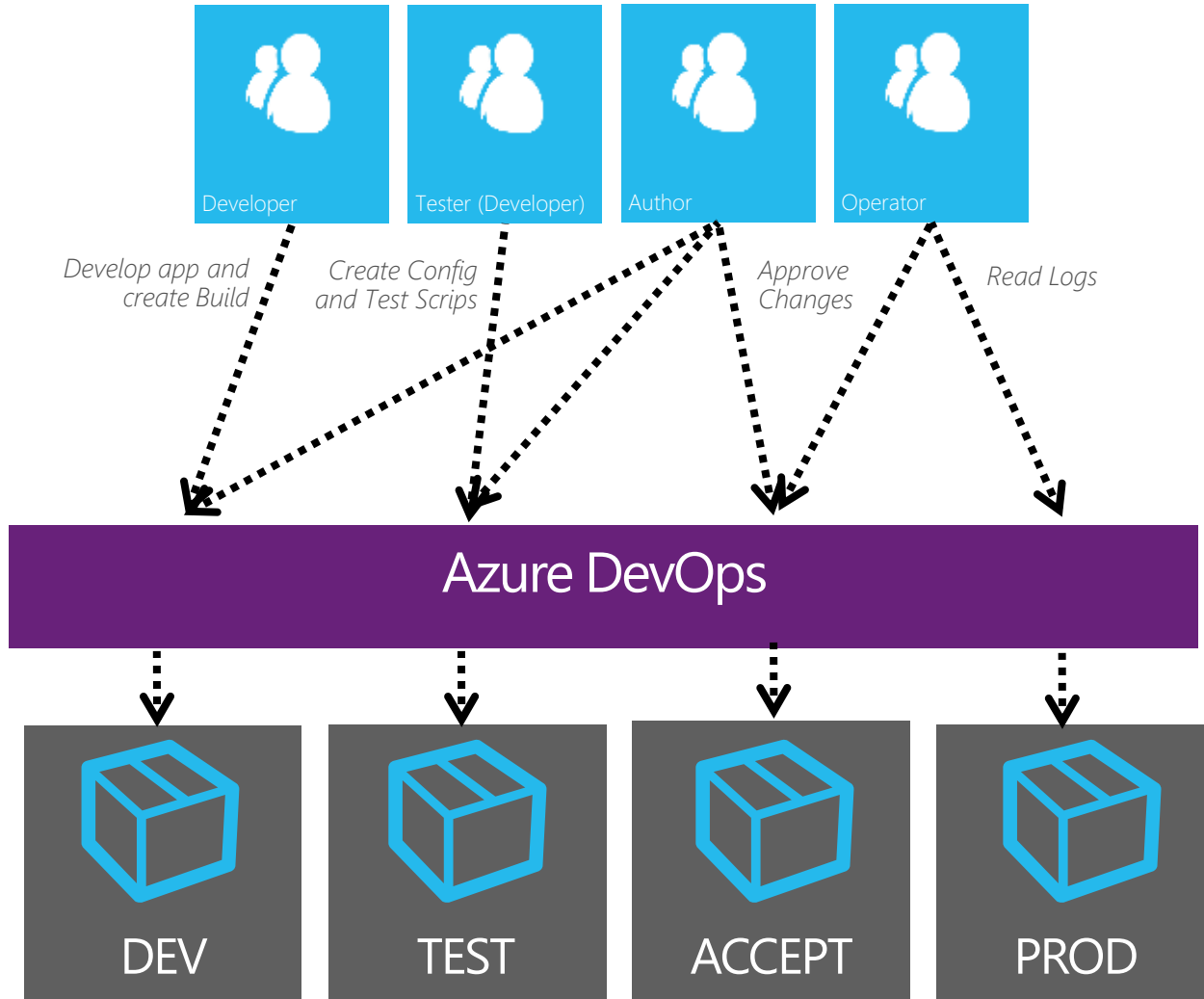


Role Based Access Control

- ✓ Allows secure access with granular permissions to resources
- ✓ Assignable to users, groups or service principals
- ✓ Built-in roles make it easy to get started



Release Automation | RBAC



RBAC in Azure DevOps

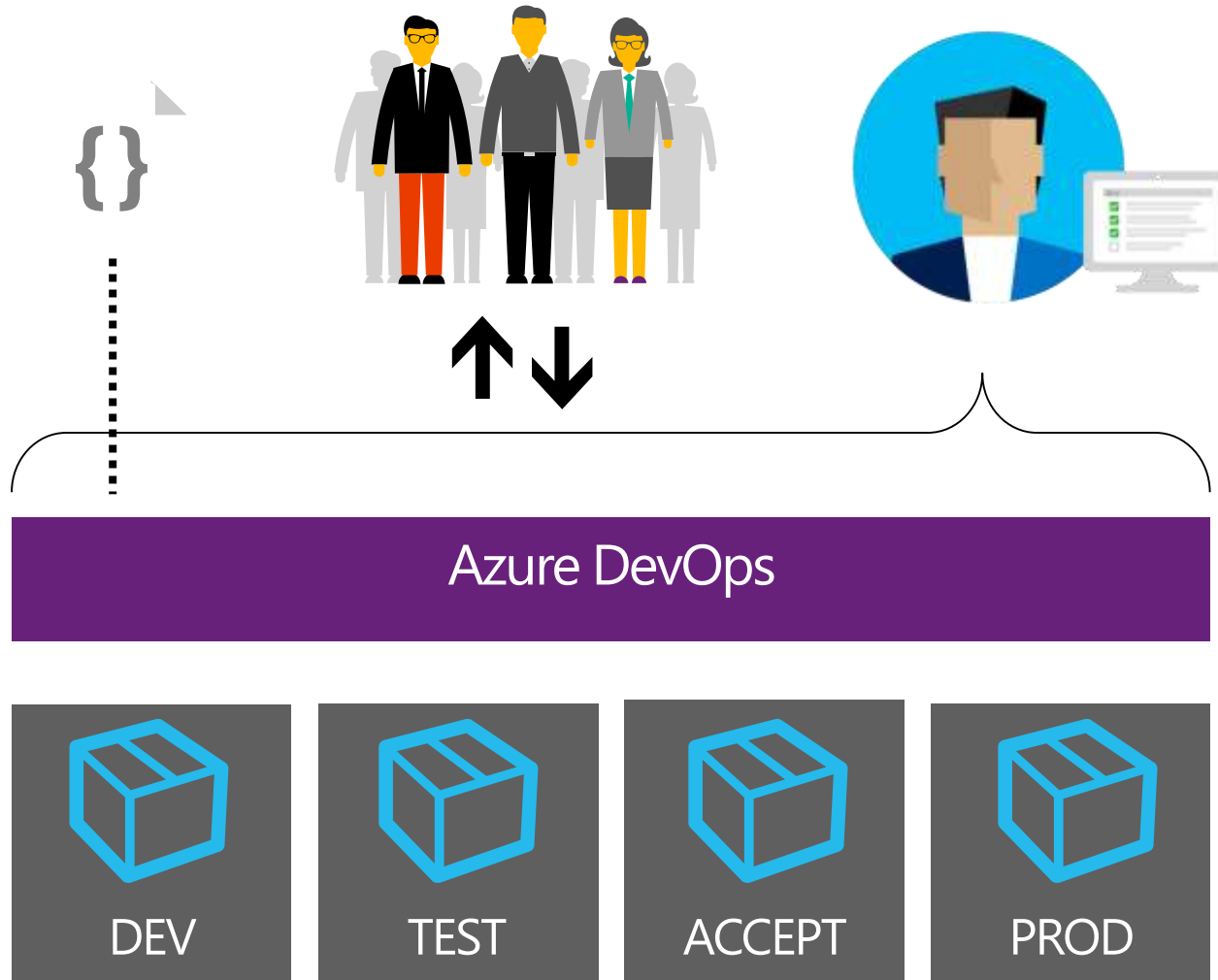


Only Azure Pipelines have access to Azure to deploy



RBAC in Microsoft Azure

Release Pipeline



Automated Deployments



Deploy the same way to all stages



Automate the approval workflow

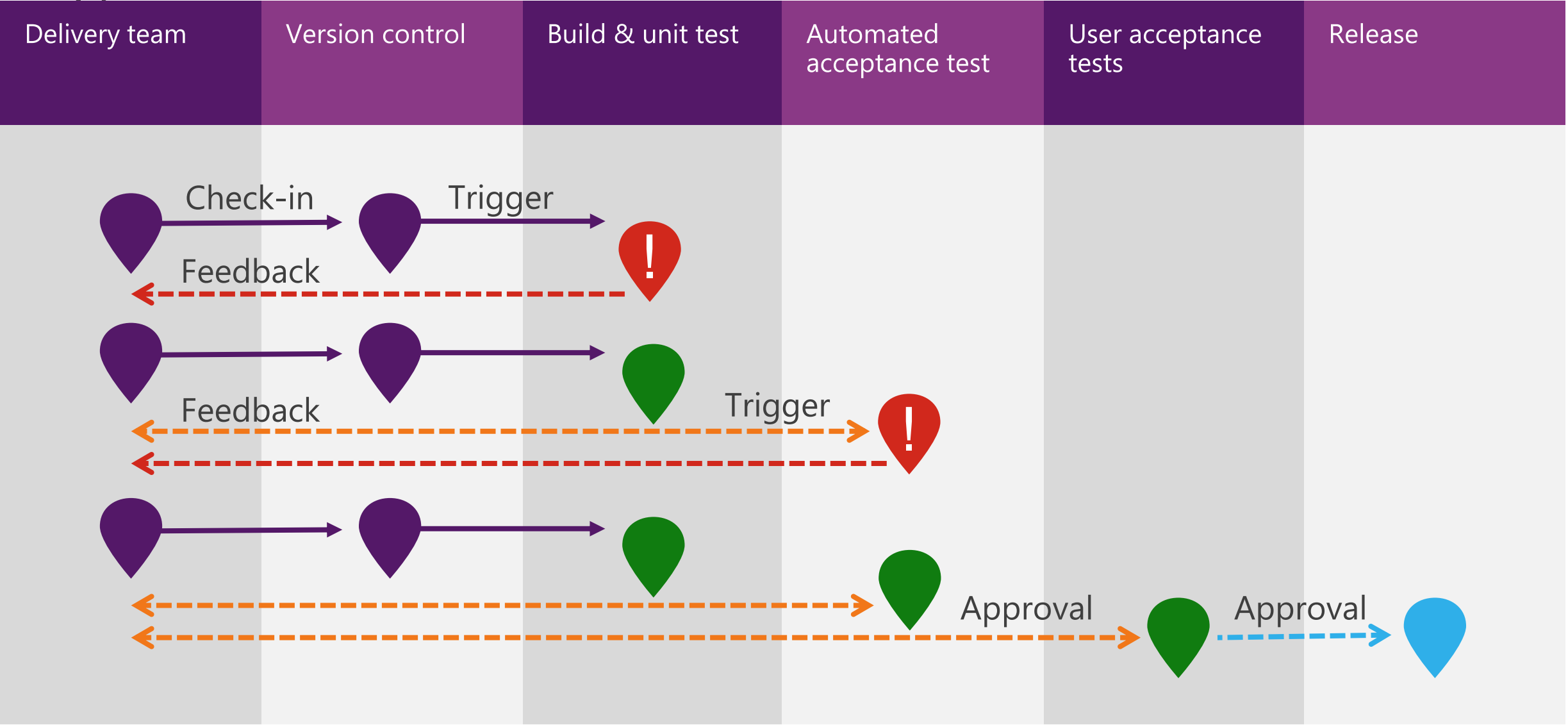


Role based access & limited standing administrative rights

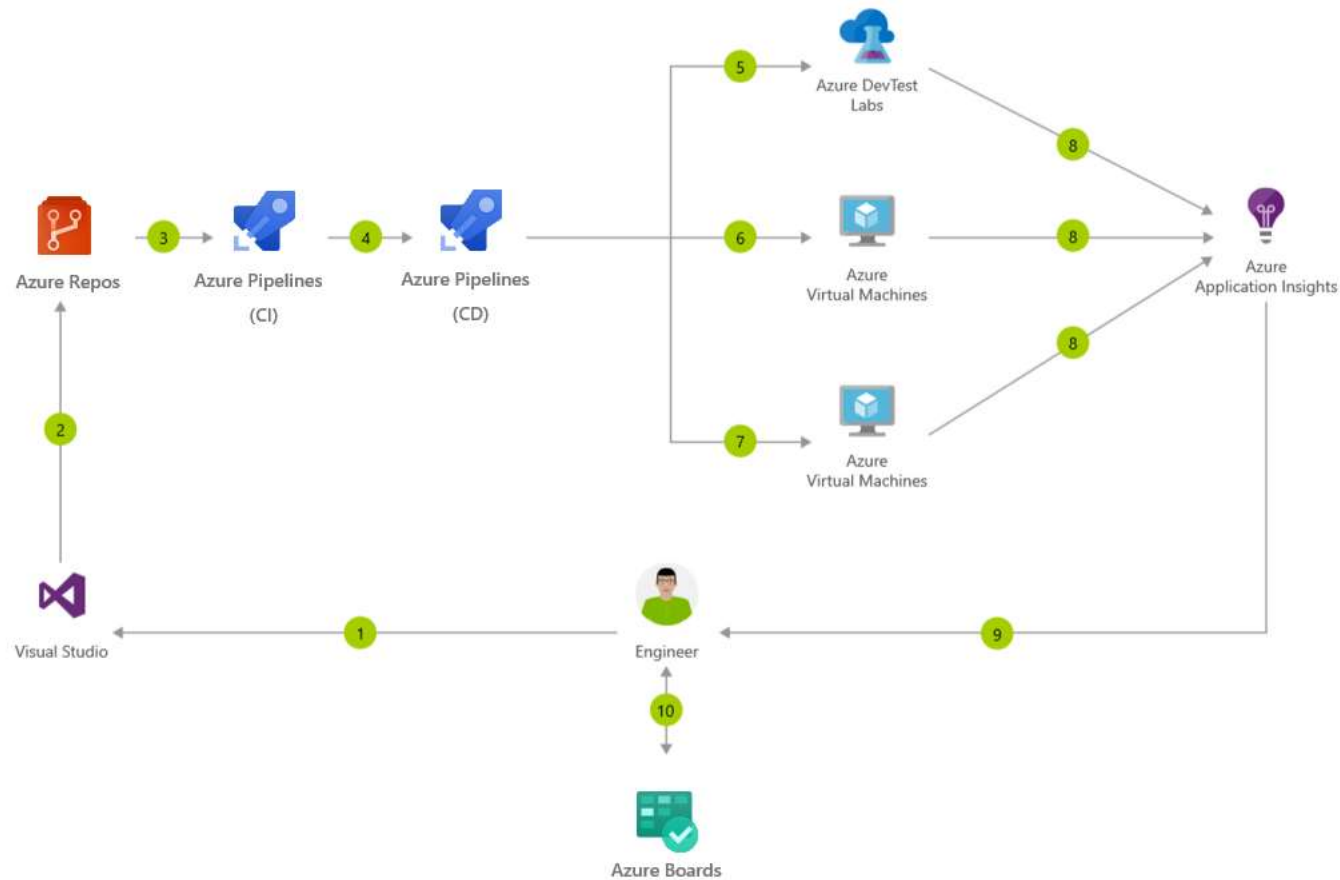


Full Traceability

Automated Release Pipeline (app & configuration)



CI/CD for Azure Virtual Machines



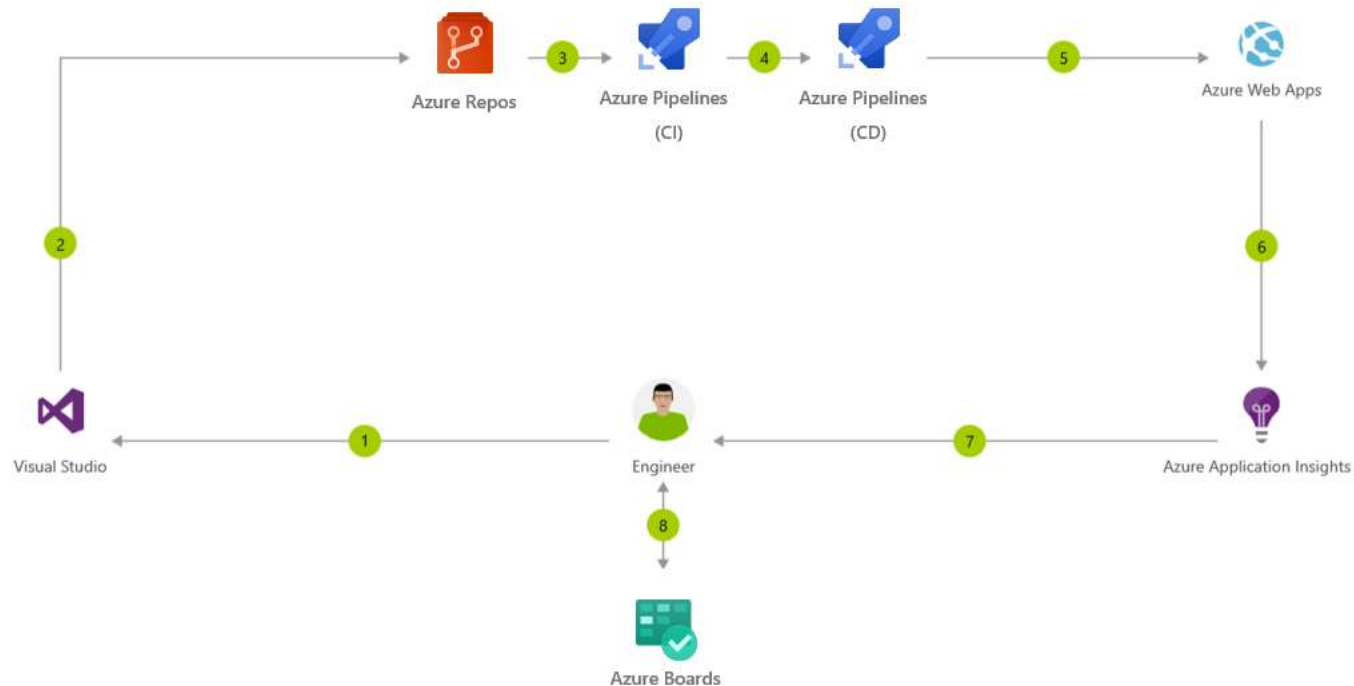
Architecture overview

Azure is a world-class cloud for hosting virtual machines running Windows or Linux. Whether you use ASP.NET, Java, Node.js, or PHP to develop applications, you'll need a continuous integration and continuous deployment (CI/CD) pipeline to push changes to these virtual machines automatically.

Azure DevOps provides the CI/CD pipeline, starting with a Git repository for managing your application source code and infrastructure code (Azure Resource Manager templates), a Build system for producing packages and other build artifacts, and a Release Management system for setting up a pipeline to deploy your changes through dev, test, and production environments. The pipeline uses Resource Manager templates to provision or update your infrastructure as necessary in each environment, and then deploys the updated build. You can also use Azure DevTest Labs to automatically tear down test resources that are not in use.

- 1 Change application source code.
- 2 Commit application code and Azure Resource Manager template.
- 3 Continuous integration triggers application build and unit tests.
- 4 Continuous deployment trigger orchestrates deployment of application artifacts with environment-specific parameters.
- 5 Deployment to QA environment.
- 6 Deployment to staging environment.
- 7 Deployment to production environment.
- 8 Azure Application Insights collects and analyzes health, performance, and usage data.
- 9 Review health, performance, and usage information.
- 10 Update backlog item.

CI/CD for Azure Web Apps

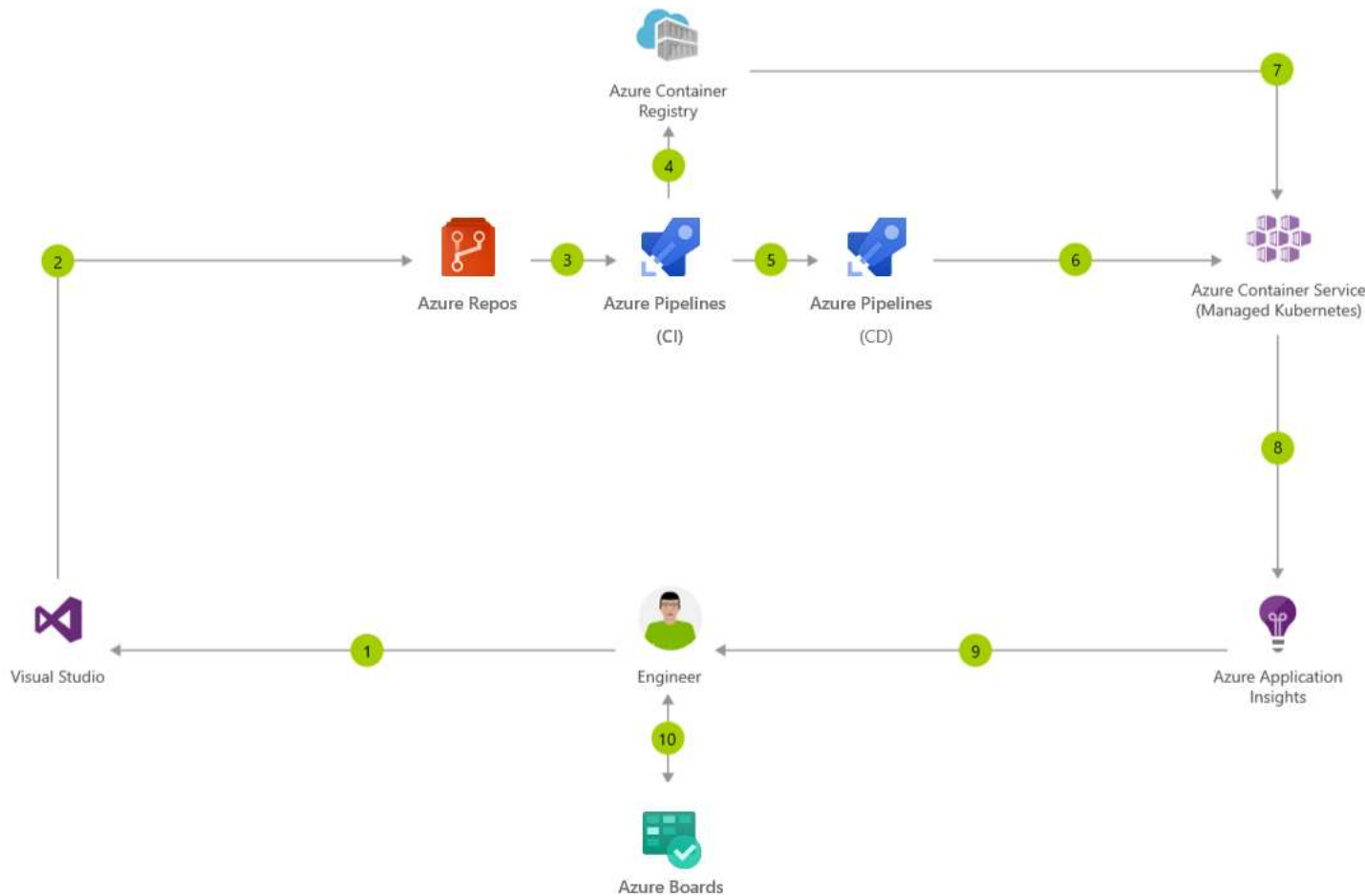


Architecture overview

Azure Web Apps is a fast and simple way to create web apps using ASP.NET, Java, Node.js, or PHP. Deliver value faster to your customers with a continuous integration and continuous deployment (CI/CD) pipeline that pushes each of your changes automatically to Web Apps.

- 1 Change application source code.
- 2 Commit application code and Web Apps web.config file.
- 3 Continuous integration triggers application build and unit tests.
- 4 Continuous deployment trigger orchestrates deployment of application artifacts with environment-specific parameters.
- 5 Deployment to Web Apps.
- 6 Azure Application Insights collects and analyzes health, performance, and usage data.
- 7 Review health, performance, and usage information.
- 8 Update backlog item.

CI/CD for Containers



Architecture overview

Containers make it easy for you to continuously build and deploy applications. By orchestrating the deployment of those containers using Kubernetes in Azure Container Service, you can achieve replicable, manageable clusters of containers.

By setting up a continuous build to produce your container images and orchestration, Azure DevOps increases the speed and reliability of your deployment.

- 1 Change application source code.
- 2 Commit Application Code.
- 3 Continuous integration triggers application build, container image build, and unit tests.
- 4 Container image pushed to Azure Container Registry.
- 5 Continuous deployment trigger orchestrates deployment of application artifacts with environment-specific parameters.
- 6 Deployment to Azure Container Service.
- 7 Container is launched using Container Image from Azure Container Registry.
- 8 Azure Application Insights collects and analyzes health, performance, and usage data.
- 9 Review health, performance, and usage information.
- 10 Update backlog item.

Configuration

- Apply DDD* when building app
 - Make app easy to configure at deployment time rather than at build
- Define variables in the release process
 - Benefits
 - Traceable
 - Secure (can be)
 - Shared
- Replace configuration on demand in pipeline

Configuration in Azure DevOps

- Use variables/variable groups to define configuration
 - Apply to release or specific environment
- Some tasks have built-in support for substitution
 - i.e Web Deploy
- Lots of options on the marketplace
 - Token replacement, regexp match/replace

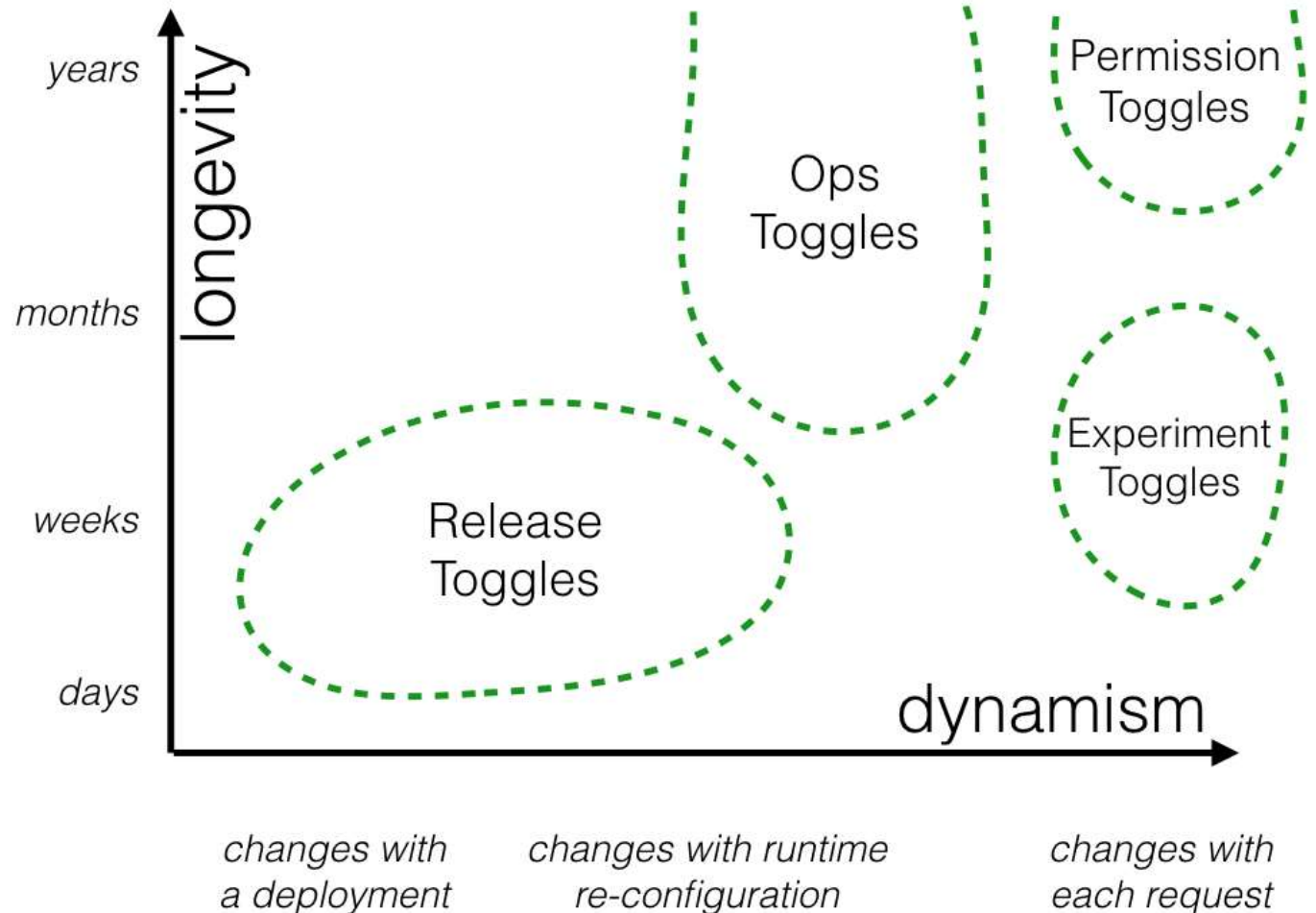
Feature toggles

- Enable features on demand
- Separates deployment from release
- Enables A/B testing
- Enabled incremental roll-out



Feature toggles

- Make sure to use the right type of feature toggles
- Ensure practices to manage toggles



Demo

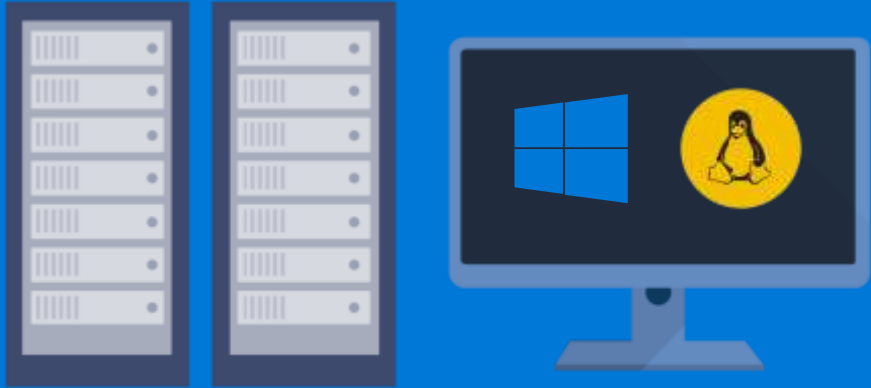
Continuous Delivery

Release Management

- From CI/CD to Release Management

Businesses want

Core-business Infrastructure & Applications



Innovation



More innovation **quicker**

Costs



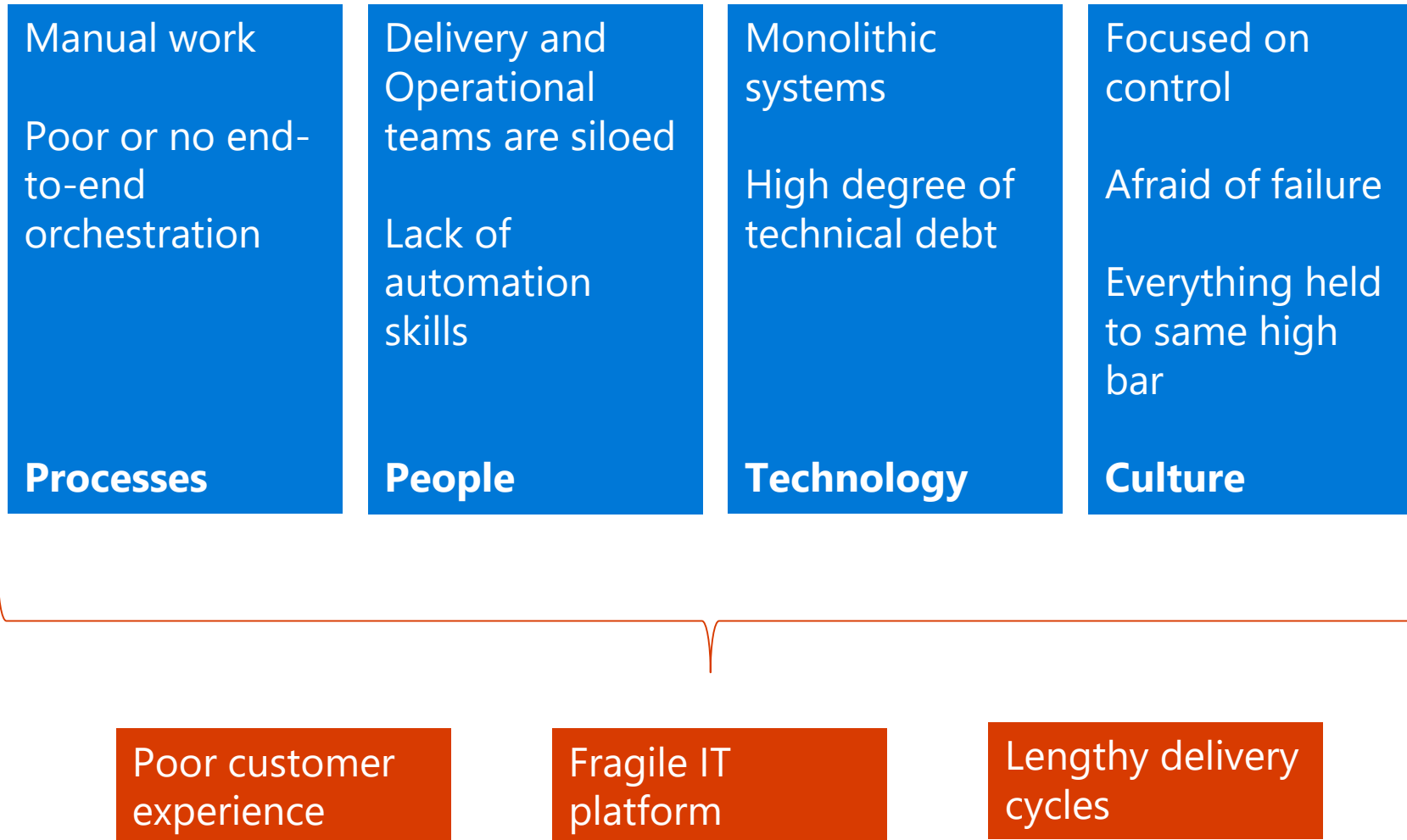
Take advantage
of (public) cloud scale
and economics

Agility



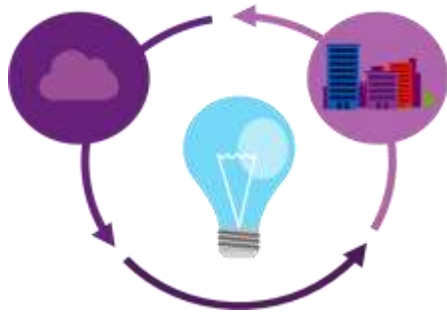
Business agility
and flexibility

What is the challenge ?



What needs to change?

Shorten cycle times
and deliver value faster



Improve quality
and availability



Optimize resources
and **eliminate waste**



Deliver **innovation** with
digital-era **velocity**



Has it been done before ?

Manual tasks combined with some automation



Completely automated / coded production process



The Release Pipeline

IS NOT

a tool
a process framework
for application code only
for bespoke applications only
DevOps

IS

people process *and* technology
a way of working
code agnostic
also usable for Commercial of the Shelf
used in DevOps

The Release Pipeline – The 4 key components



Source



Build



Testing



Release

Who changed the environment?

(control who can make code changes)

What did they change, exactly?

(record detailed code changes)

When did the change occur?

(track history of code changes)



What we did before

- Prototype in VM
- Submit change form
- Present to CAB
- Argue that the change is worthwhile
- Conflict oversight by committee
- Update docs



What we do now



- Document everything in configuration as code
 - Including automated deployment and testing
- Clone latest code to workstation
- Make code changes
- Run tests locally
- Push to Source
 - Who, What, When, Why (comments)
- Merge with Master, resolve conflicts if needed

How will I catch problems at the earliest possible moment?

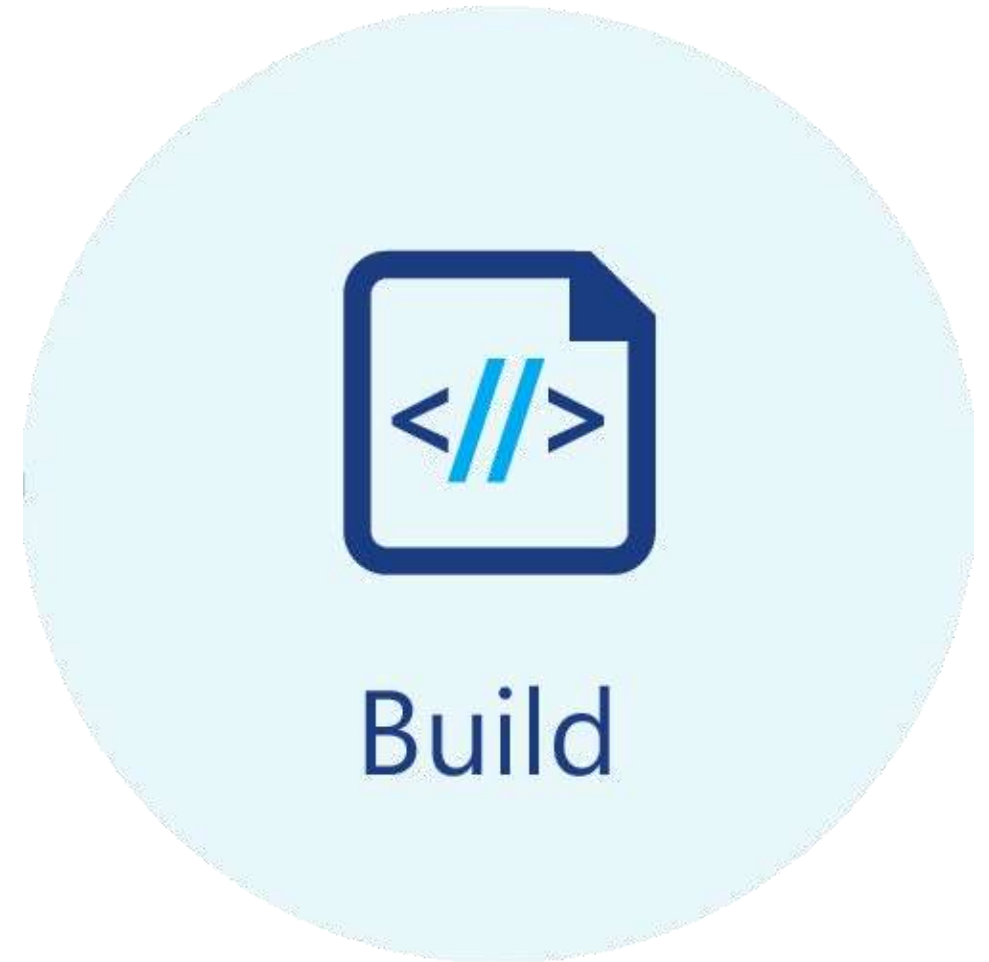
(run scripts that perform all actions that must occur before a change can be released into production)

Can elements be combined cleanly to produce the correct results?

(implement user-defined automated processes when certain changes are made to files in source control)

How will I be notified of a problem?

(monitor successful implementation of automated processes)



What we did before



- Remote in on Saturday
- Make sure servers are listed in RDCMan
- Connect to servers and make changes
- Reboot in order
- Say a prayer / squeeze rabbit's foot / perform tribal dance
- Check that services still work correctly

What we do now

- Trigger Build service
- Run scripts stored in Source
- Validate quality
- Perform work
- Capture output



How do we check for regulatory issues?

How do I know this change will not cause an outage?

Will this change work across every variation I have in my environments?

Does this configuration meet our business requirements?



Testing

What we did before



- Create VM
- Install OS
- Install app/service
- Implement change
 - Manual test, run script
- Try app/service to make sure it still works

What we do now



- Build runs Scripts from Source
- Script Analyzer (aka Linting)
 - Must meet guidelines set by organization
- Unit Tests (e.g. using Pester)
 - Functionally should work as expected
- Integration Tests (e.g. using Pester / Kitchen)
 - Should work across matrix of diverse combinations
- Operational Validation (e.g using Pester / OVF)
 - Service should do what it is supposed to do
- TDD

How do I make changes *without* granting *long term administrative access*?

Does anyone need to sign-off / *approve before deployment* (outside the team)?

How do I *keep services consistent* across all my environments?

Can I integrate service management?



What we did before



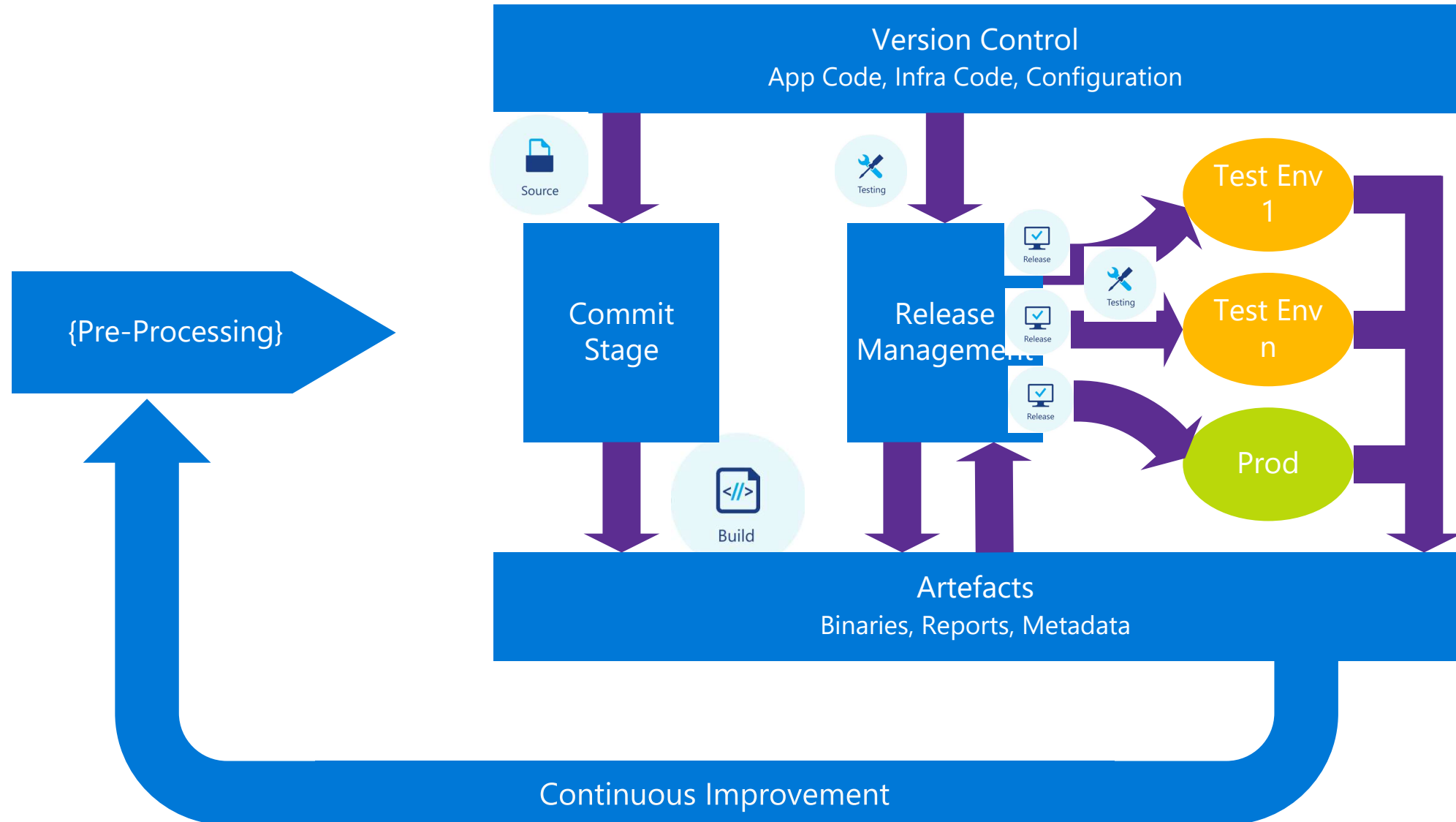
- Request/create VM
- Request/manage environmental changes
- Install OS
- Install app/service (or defer to app/service owner)
- Onboard to patching, anti-malware, backup, monitoring
- Validate app/service is working
- Go Live

What we do now



- Build runs scripts from Source
- Potentially automatically releases when Tests pass
- Deploy artifacts to environment
- Simplify complex work through automation
- Promote through stages
 - QA, Prod, Green/Blue
- Enable features on demand

Release Pipeline Logical Architecture



Demo

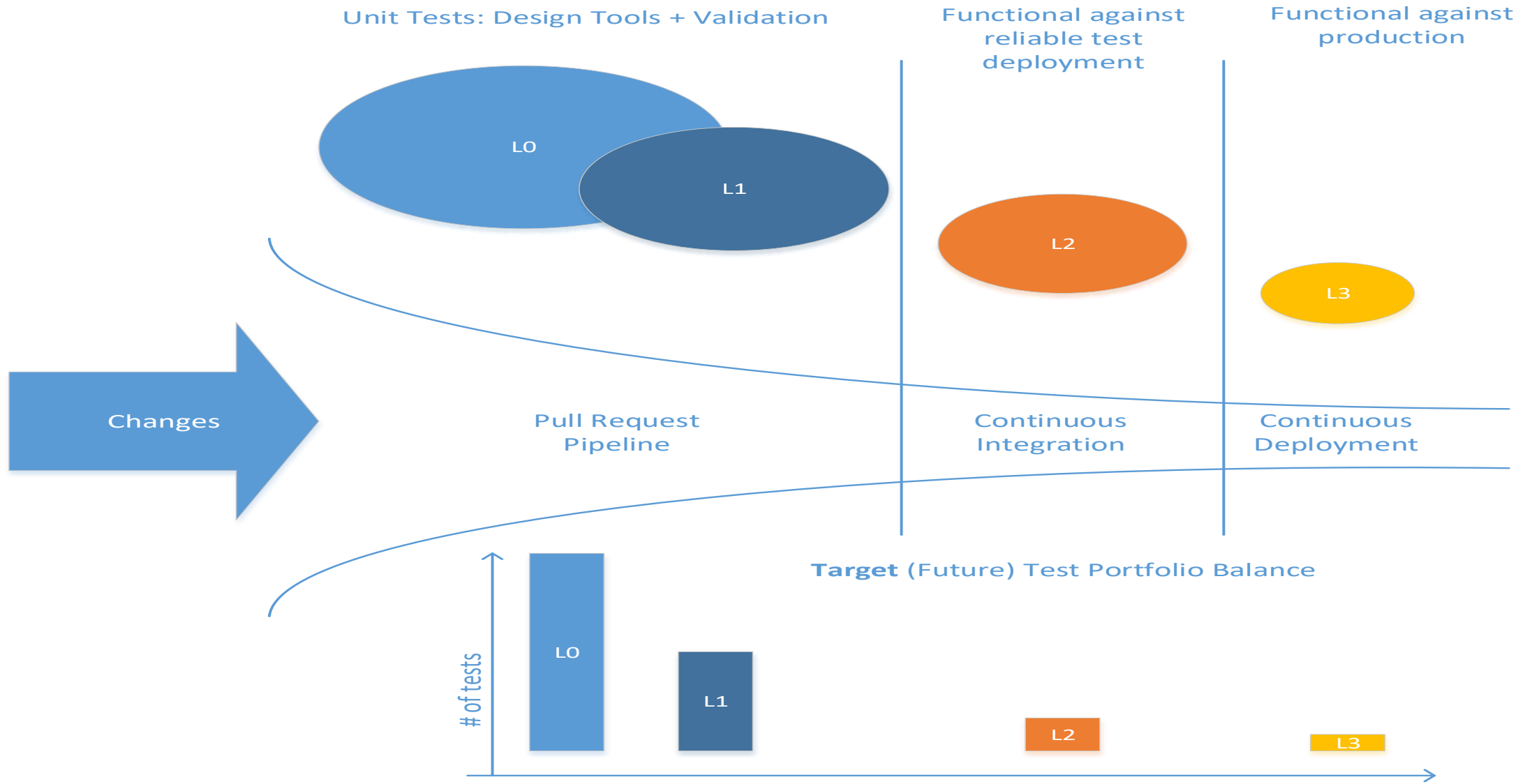
Release Management

Adding testing

Automated testing

- Faster releases require automation
- Should be done as a team
- Run automated tests as early as possible
 - “Shift-left”

"Shift-Left" == Pushing Quality Upstream










Tasks for testing

- Visual Studio Test for tests with a VS test adapter
- Any tool + Publish Test Results for custom tools
- VS Test Platform Installer helps getting the test tools to new servers

Add tasks | Refresh

All Build Utility **Test** Package Deploy Tool Marketplace

-  **App Center Test**
Test app packages with Visual Studio App Center.
-  **Cloud-based Apache JMeter Load Test**
Runs the Apache JMeter load test in cloud
-  **Cloud-based Load Test**
Runs the load test in the cloud with Visual Studio Team Services
-  **Cloud-based Web Performance Test**
Runs a quick web performance test in the cloud with Visual Studio Team Services
-  **Publish Code Coverage Results**
Publish Cobertura or JaCoCo code coverage results from a build
-  **Publish Test Results**
Publish Test Results to VSTS/TFS
-  **Visual Studio Test**
Run unit and functional tests (Selenium, Appium, Coded UI test, etc.) using the Visual Studio Test (VsTest) runner. Test frameworks that have a Visual Studio test adapter such as MsTest, xUnit, NUnit, Chutzpah (for JavaScript tests using QUnit, Mocha and Jasmine), etc. can be run. Tests can be distributed on multiple agents using this task (version 2).

Integrating with the pipeline

- Just like apps tests need to adapt to the environment
- Pass context from test task

```
<?xml version="1.0" encoding="utf-8"?>
<RunSettings>
  <!-- Parameters used by tests at runtime -->
  <TestRunParameters>
    <Parameter name="webAppUrl"
      value="http://localhost:5000/" />
  </TestRunParameters>
</RunSettings>
```

Settings file ⓘ

\$(System.DefaultWorkingDirectory)/_MyHealthClinic-UITest/drop/src/MyHealth.Web.UITests/bin/Release/default.runsettings

Override test run parameters ⓘ

-webAppUrl \$(webAppUrl)

```
[TestInitialize()]
public void SetupTest()
{
    appURL = TestContext.Properties["webAppUrl"].ToString();
}
```

Platform services

- Pipeline agent also runs tests
- Hosted agent can now run interactive tests
- Make sure to install dependencies
 - Drivers
 - Browsers
 - ...

Self-service dev/test environments

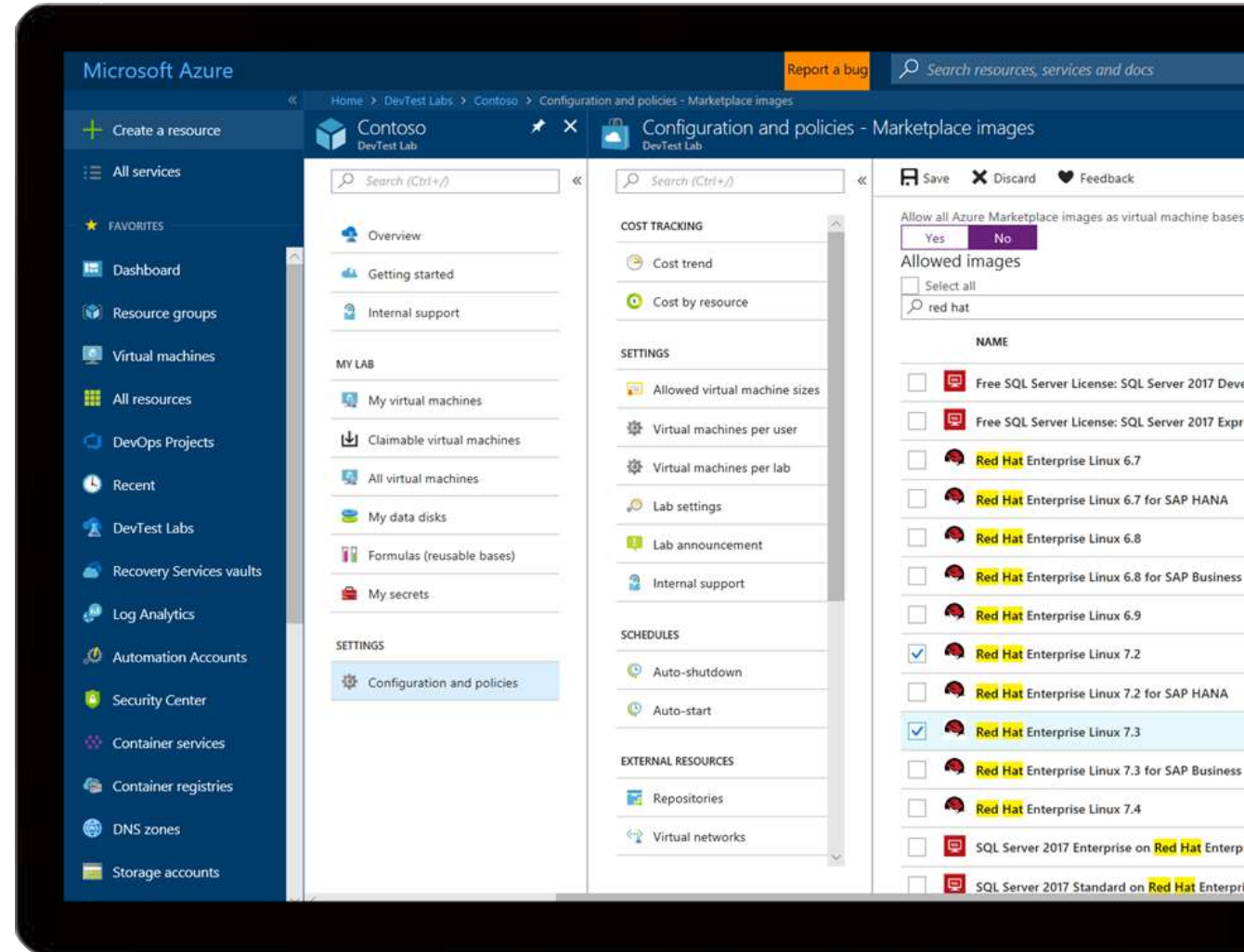
Azure Lab Services

Simplify cloud environment management for developers and testers.

Enforce policies and control costs with full visibility

Use templates, custom images and formulas to reproduce environments.

Orchestrate with Azure Pipelines or integrate using REST API



Demo

Automated testing

Adding telemetry

Measure, monitor and improve

- Improve process and application by continuous monitoring
- Azure operational and usage telemetry
- Azure DevOps Dashboards

Smart insights, faster

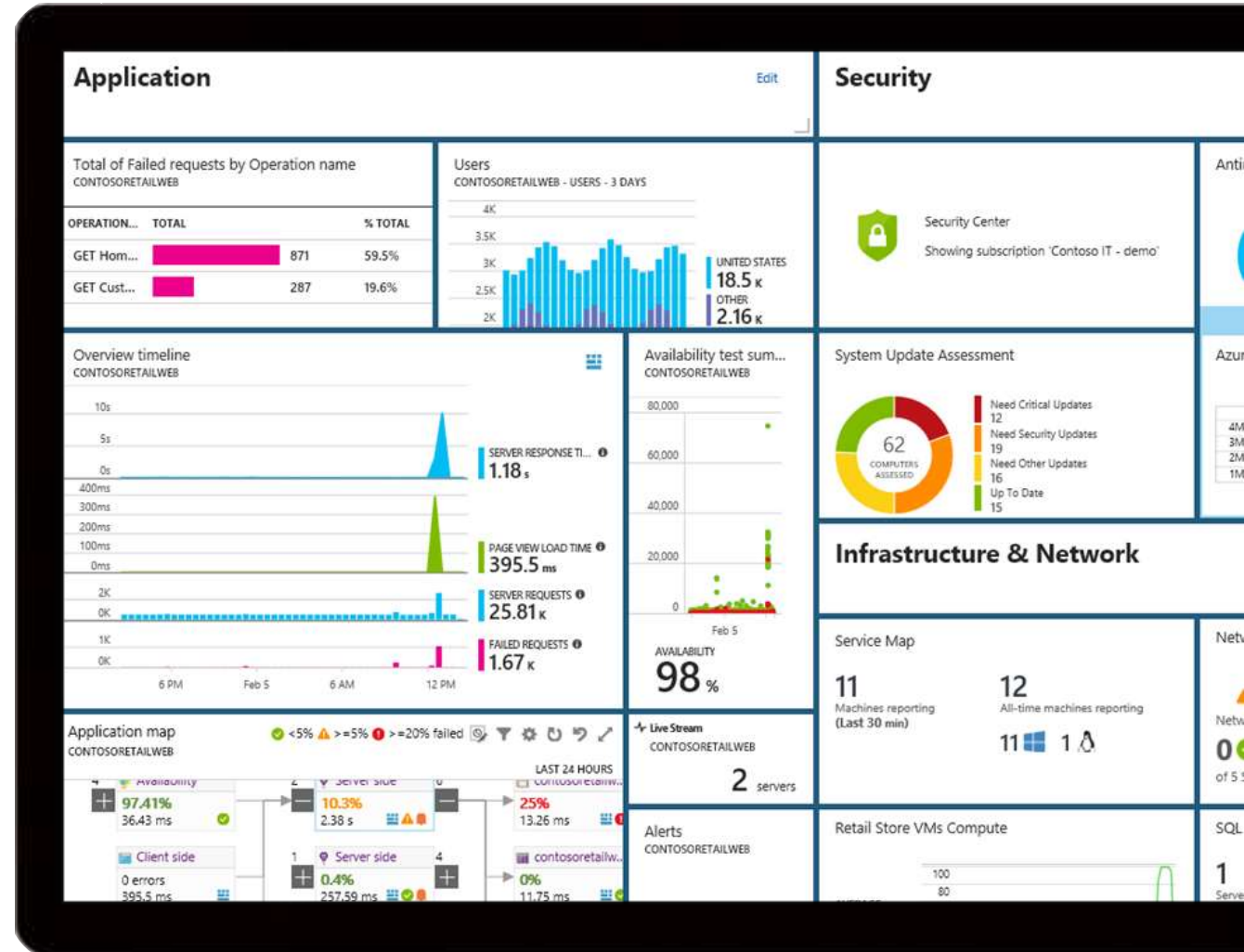
Azure Monitor, Log Analytics & Application Insights

Pre-defined solutions with smart thresholds

Visualize data in intuitive and customizable dashboards

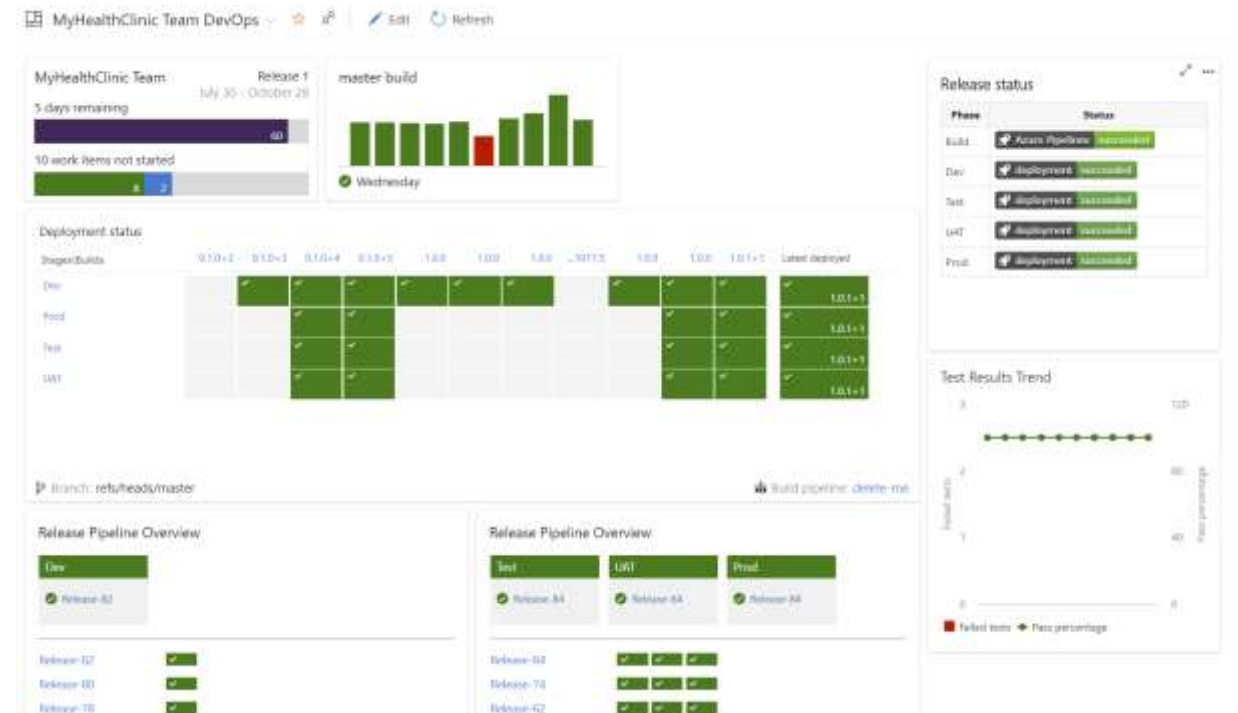
Separate the signal from the noise and accelerate root-cause analysis

Integrate your existing processes & tools like Service Now



Azure DevOps Dashboards

- Insights into the release process
- Add widgets for build, release, test status



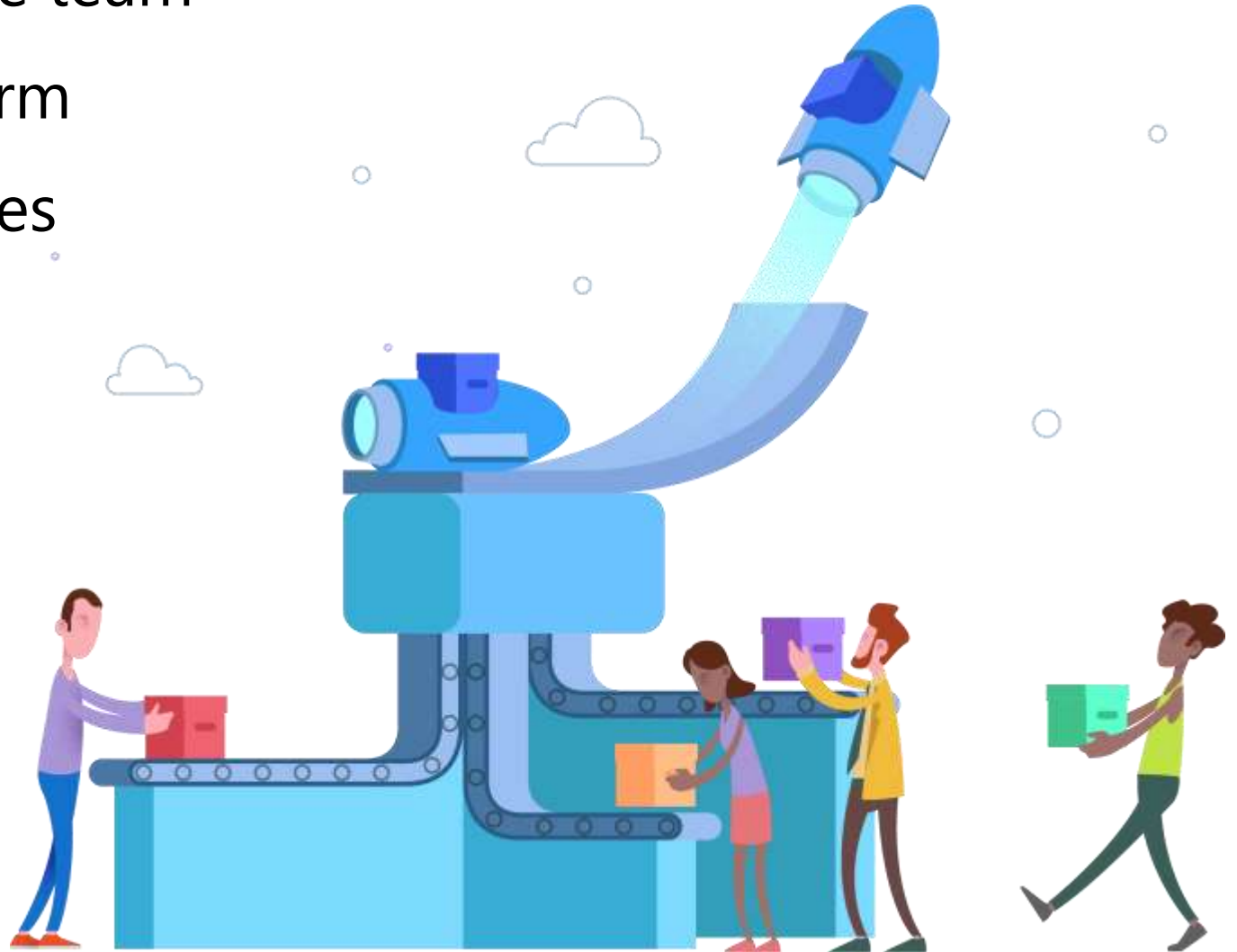
Demo

Dashboards

Wrap-up

Azure DevOps

- Collaboration platform for the team
- Great for any tool, any platform
- Great for any modern practices



Sessions

DevOps with Azure, Kubernetes, and Helm

2018-10-24 11:00 – 12:00

Keynote room

In this session, we will show you how you can use standard DevOps practices such as IaC, CI/CD, automated release and more in conjunction with Kubernetes (AKS) and Helm. For those who don't know, Helm is a tool that streamlines...



Jessica Deen

Defining the OPS in DEVOPS

2018-10-25 14:00 – 15:00

E6-E7

For many organizations, delivering software into production has become increasingly more complex with long testing cycles and a division between development and operations teams. DevOps is a cultural movement that is breaking down those barriers. Focusing on automation, collaboration, tools,...



Olov Norman

Continuous Delivery med Azure DevOps Pipelines

2018-10-25 12:30 – 13:30

M5

I denna session får du lära dig hur du går från kod till produktion (och tillbaka igen) genom att tillämpa goda principer inom continuous delivery. Vi kommer att ta en modern applikation, paketera och publicera den till miljöer för test...



Mathias Glaussou

Get started with containers in Azure

2018-10-24 17:30 – 18:30

E4

Containers are the next stage in the virtualization of computing resources. This session gives you a deep understanding of containers, docker, how to deploy containers on Azure VMs and an overview of container-clustering using Kubernetes.



Daniel Bugday

Introduction to Azure Kubernetes Services

2018-10-24 13:15 – 14:15

M8

Containers are becoming more popular every day, and for good reasons. Containers bring a lot of power both for developers and operations, solving the growing problem of how to efficiently design, deploy and monitor applications written in many different languages...



Jakob Ehn

Consuming cloud services with the Kubernetes Service Catalog

2018-10-25 10:30 – 11:30

M6

In a cloud native world, managed services such as database, storage, and event processing systems can be utilized by applications without the overhead of total service ownership. Kubernetes provides an extension mechanism for dynamically requesting and consuming managed services through...



Neil Peterson

Building secure infrastructure in Azure with Terraform, Ansible and Powershell DSC

2018-10-25 09:00 – 10:00

M4

Terraform gains attention as one of the leading open-source tools for delivering cloud infrastructure. Enhanced with powers of Ansible and Powershell DSC in Azure, it allows to control the state of machines and keep them secure according to the defined...



Igor Andriushchenko

Managing Secrets in modern application development

2018-10-24 13:15 – 14:15

M4

Managing Secrets in modern application development requires more effort than storing clear text password, connection strings and api keys in your configuration files. It requires storing secrets in secure location outside of source control and managing the set of secrets...



Taavi Koosaar

Resources

Azure DevOps

<https://azure.microsoft.com/en-us/services/devops/>

Azure Pipelines

<https://azure.microsoft.com/en-us/services/devops/pipelines/>

Hands-on labs

<https://www.azuredevopslabs.com/>

Thank you!

Please evaluate my session in
the TechDays app!

