Microsoft

Microsoft Azure Virtual Training Day: DevOps with GitHub

Microsoft

Getting Started with DevSecOps

DevOps Learning Path

- **Getting Started with DevOps**
- Managing the Flow of Work
- Shift security “left” in your CI/CD process
- Delivering changes to cloud
- Performance Monitoring and maintenance

TAILWIND TRADERS

Who is Tailwind Traders?

Experiencing rapid growth

Differing goals across teams

Need for better collaboration

Looking to Implement

- DevOps methodology
- Better communication tools
- Shared tooling

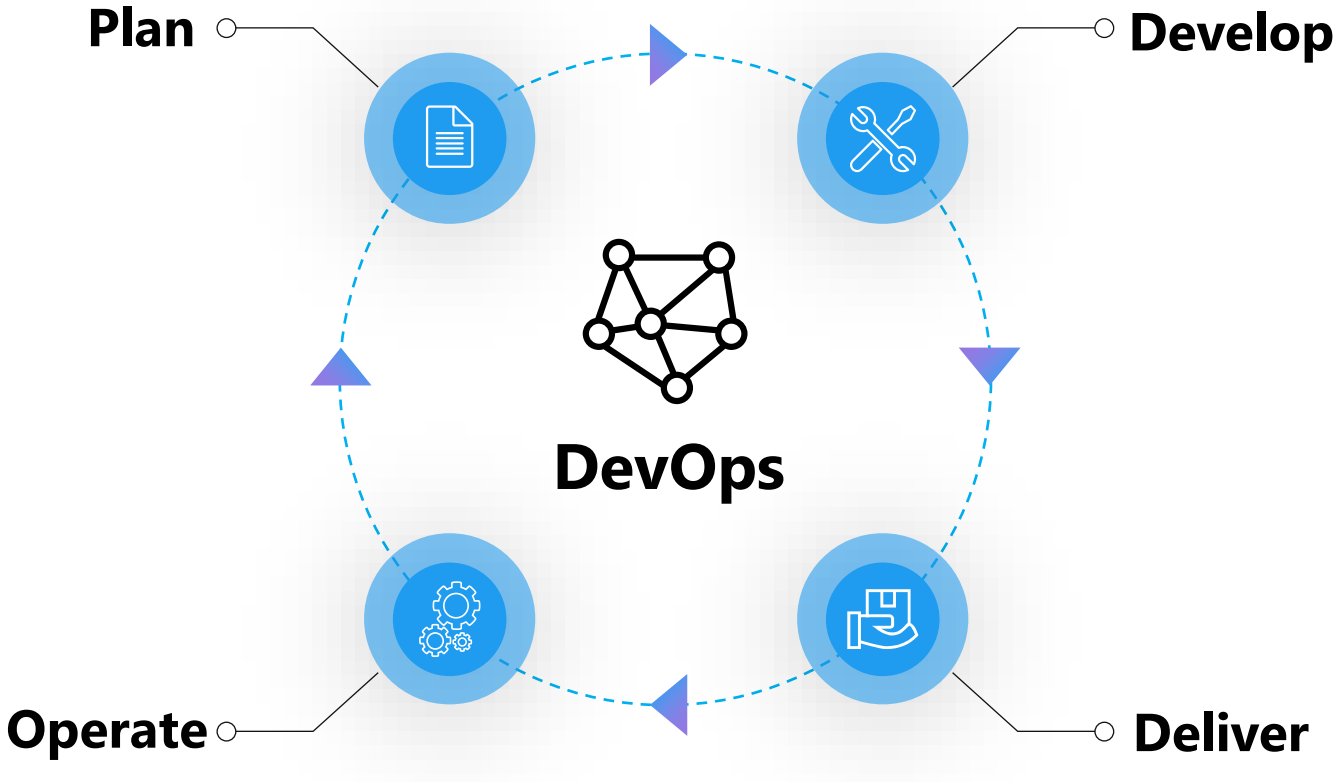
What is DevSecOps?

How DevOps evolved into DevSecOps

DevOps accelerates delivery

> *DevOps is the union of **people**, **process**, and **products** to enable continuous delivery of value to your end users.*
>
> – **Donovan Brown**

Learn about What is DevOps

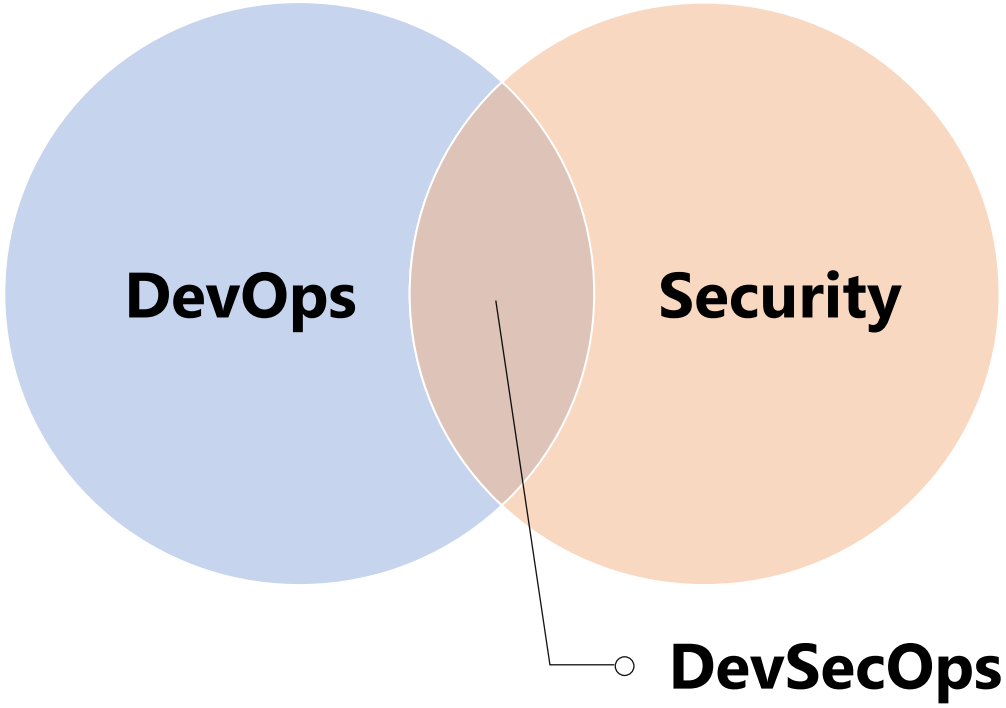


DevSecOps makes delivery secure

> *DevSecOps is the **process** of **integrating** and **automating** preventive, detective, and responsive **security** controls into the pipeline.*

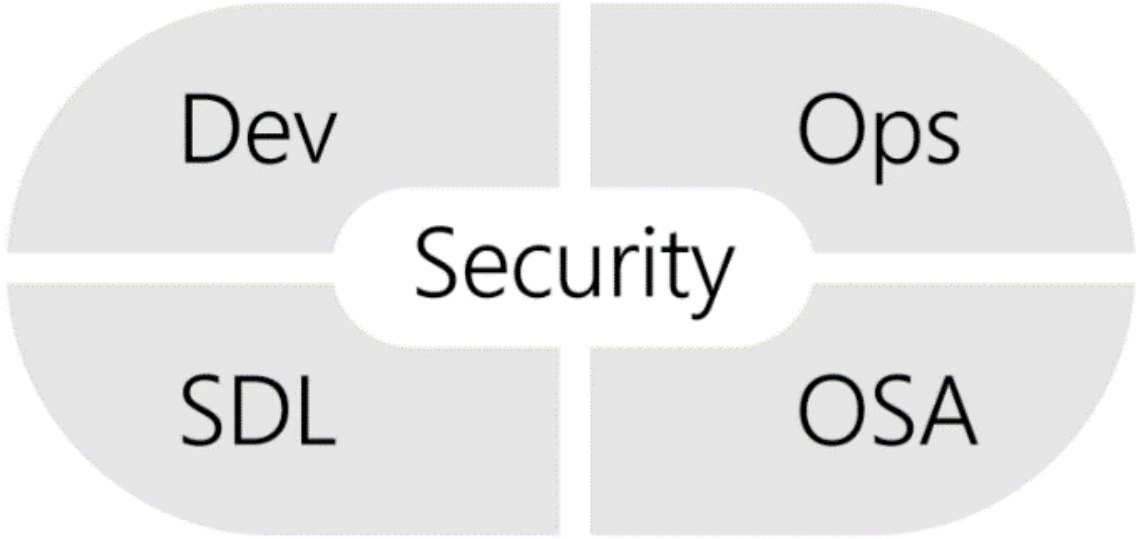
Learn about DevSecOps

It's a union of security, development, and operations teams.

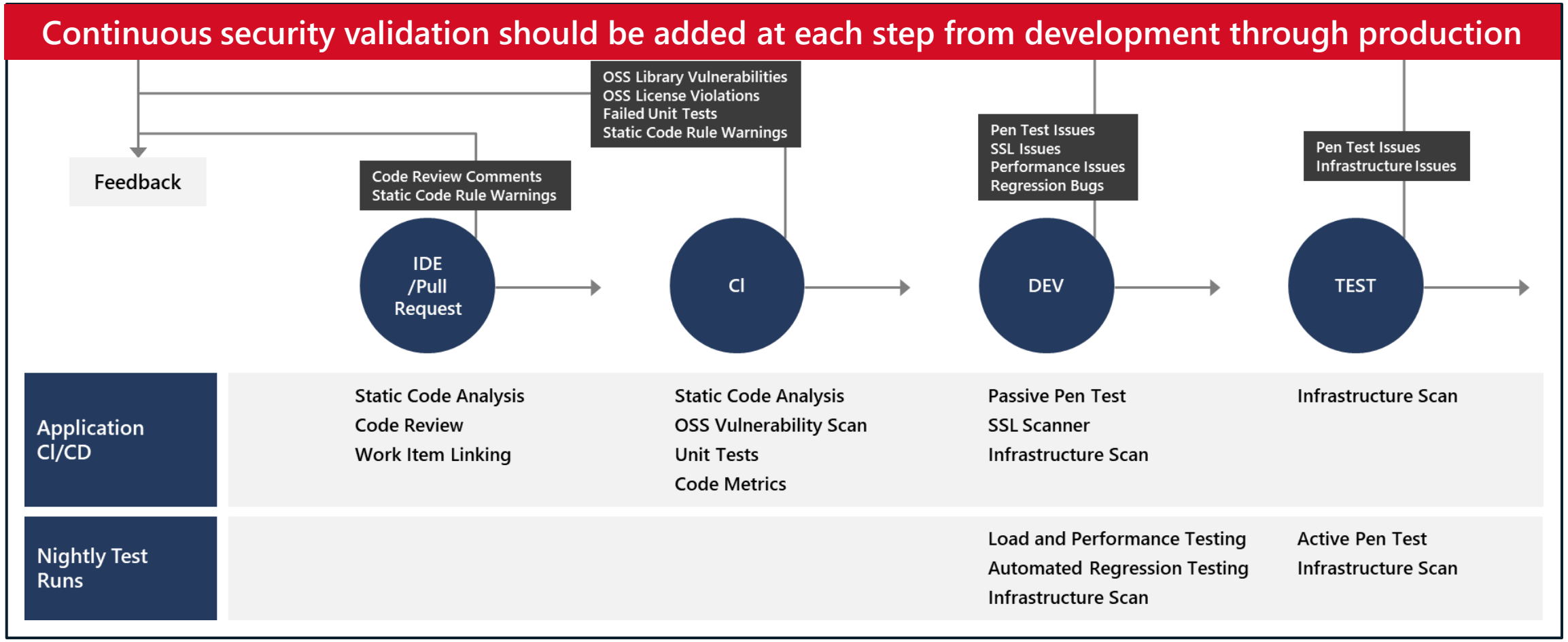


Benefits of DevSecOps

- Shift their security left to occur at more critical points throughout the development lifecycle, aiding in lower vulnerability remediation time.
- It also helps organizations to form a seamless workflow by integrating it into existing toolchains.
- More so, this aids organization in continually identifying new threat vectors.

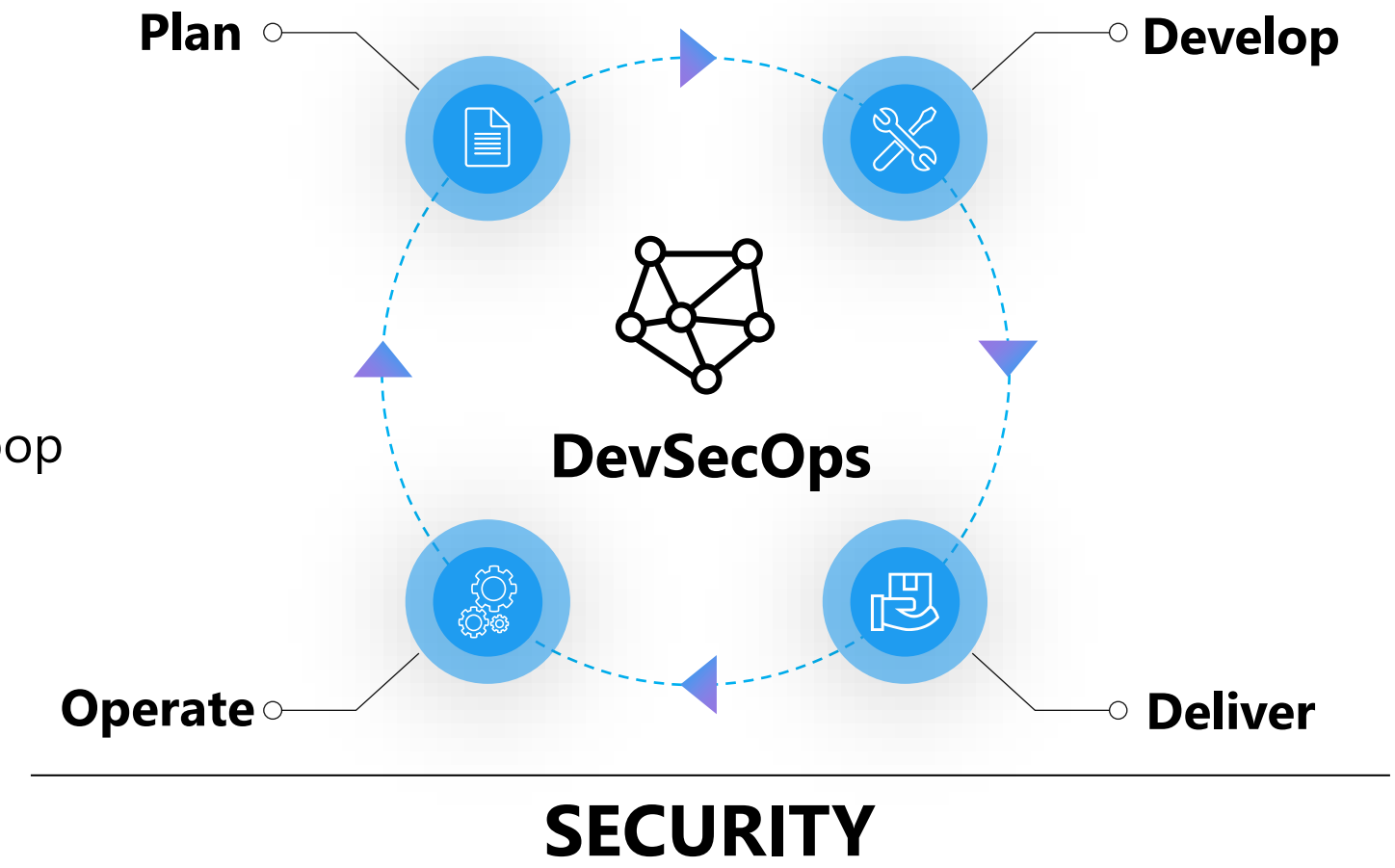


DevSecOps key validation points

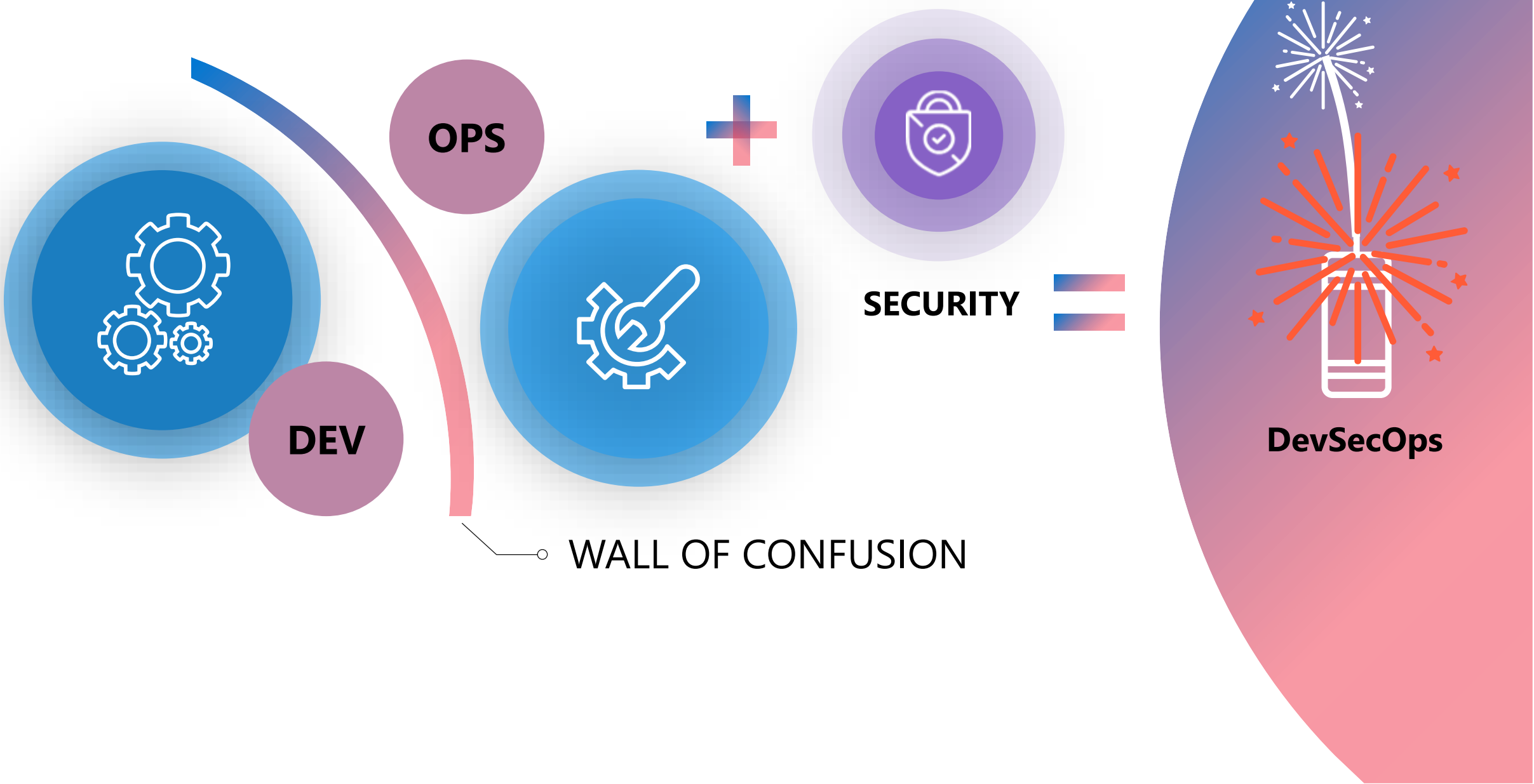


What does this all mean?

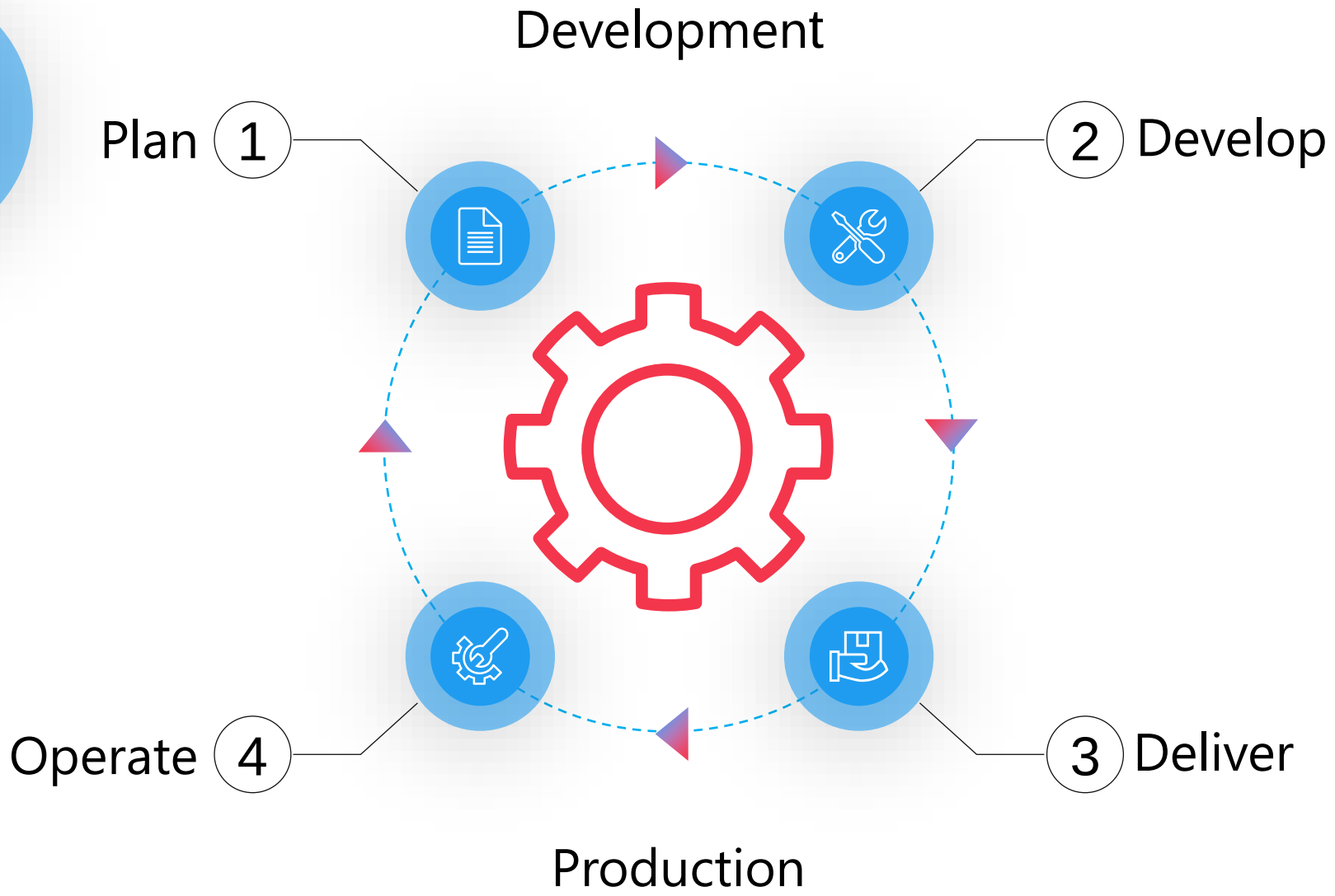
- Deliver value
- Increasing efficiency
- Eliminating waste
- Streamline the feedback loop
- Continuously improve
- Deliver faster
- Happier and secure customers



People



Process

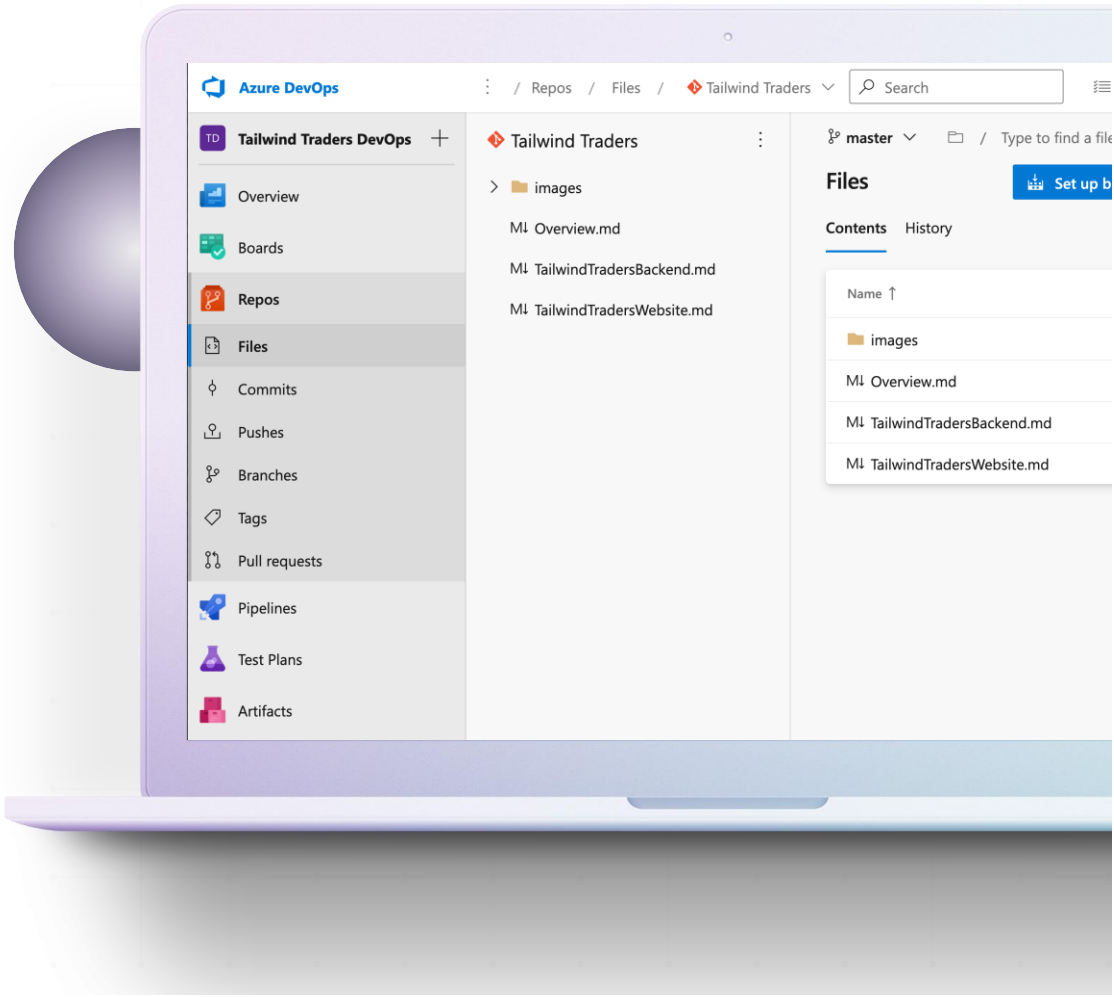

Development
Plan
1
2
Develop
Operate
4
3
Deliver
Production

How does GitHub play a role in the DevSecOps process?

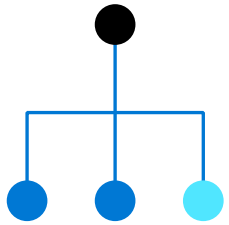
Introduction to Git and GitHub

What is Source Control and Why do we Need it?

- A form of version control
- Uses concept of code repositories
- Tracks changes made within repositories
- Allows for cross-team collaboration
- GitHub



Understand distributed source control



Distributed

Strengths	Best Used for
• Cross platform support • An open-source friendly code review model via pull requests • Complete offline support • Portable history • An enthusiastic growing user base	• Smaller size (in bytes) and modular codebases • Evolving through open-source • Highly distributed teams • Teams working across platforms • Greenfield codebases

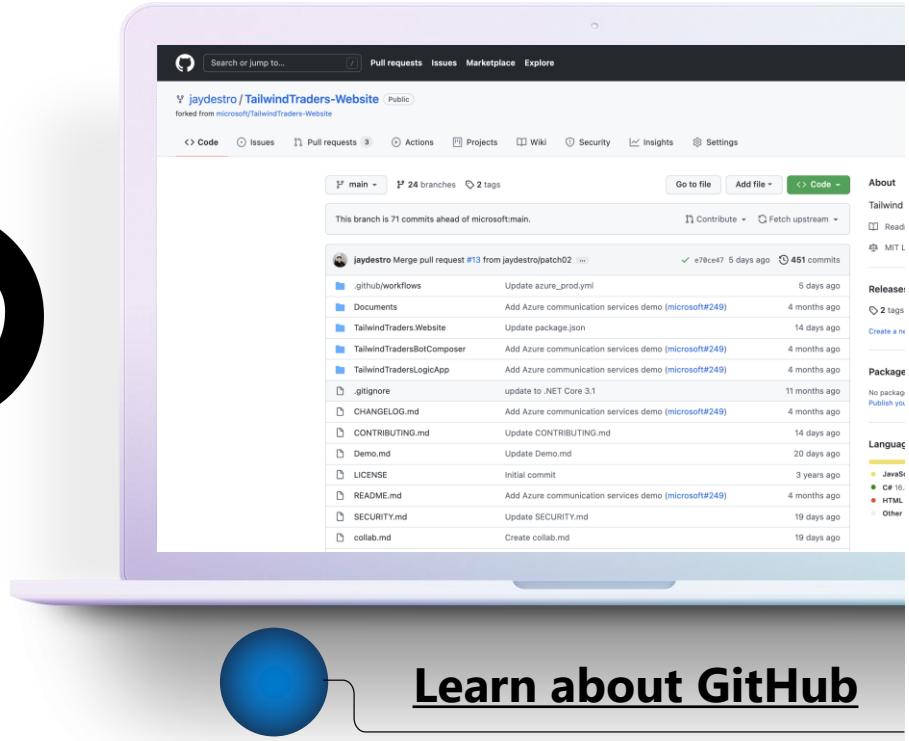
Every developer clones a copy of a repository and has the full history of the project.

Common distributed source control systems are Mercurial, Git, and Bazaar.

What is GitHub?

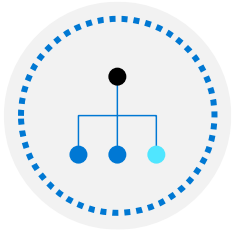
GitHub is the leader in Git repository hosting. Some key features of GitHub

- Expertise sharing
- Cross-team collaboration
- Improved code reuse
- Codespaces on GitHub
- GitHub Actions (CI/CD)
- Increased velocity



Learn about GitHub

GitHub benefits in the DevSecOps culture



Largest open-source community

Features of GitHub:

- Automate from code to cloud
- Securing software, together
- Seamless code review
- Code and documentation in one place
- Coordinate
- Manage teams

Components of a Git Project

Code!

Branches – isolate development work

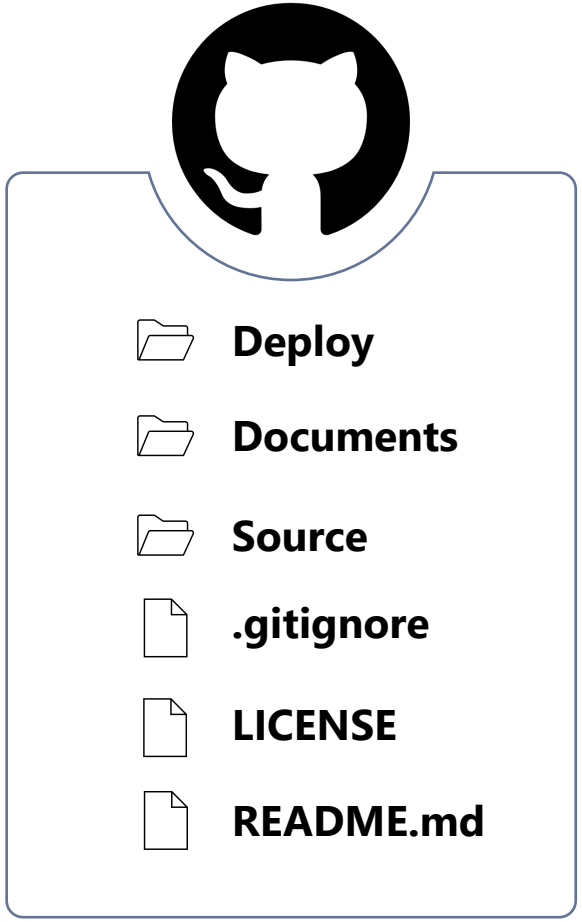
Tags – point to a specific release

.gitignore – untracked files to ignore

Commits – track changes to artifacts

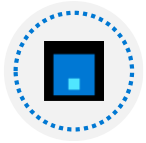
GitHub Project files

- README.md file – Document your project
- SECURITY.md file – Define your security policy
- LICENSE file – Define the license for your project
- CODEOWNERS file – Define who is responsible for code
- Pull Requests – Request to merge your changes
- Issues – Track issues/bugs/features
- Releases – Bundle specific iterations of your project



GitHub features

Security: Review code and identity vulnerabilities early in the development cycle



Repos: Provide cloud-hosted and on-premises git repos for both public and private projects

Actions: Create automation workflows with environment variables and customized scripts

Packages: Ease integration with numerous existing packages and open-source repositories

Codespaces: Provide cloud-hosted collaborative development environments

Copilot: Use OpenAI Codex to suggest code and functions in real-time from editor

Explore source control integration

Azure Automation supports source control integration

Easier collaboration

Increased auditing and traceability

Roll back to earlier versions of your runbooks

Can push code from Azure Automation to source control or pull your runbooks from source control to Azure Automation

Azure Automation supports the following source Control options:

- GitHub
- Azure DevOps (Git)

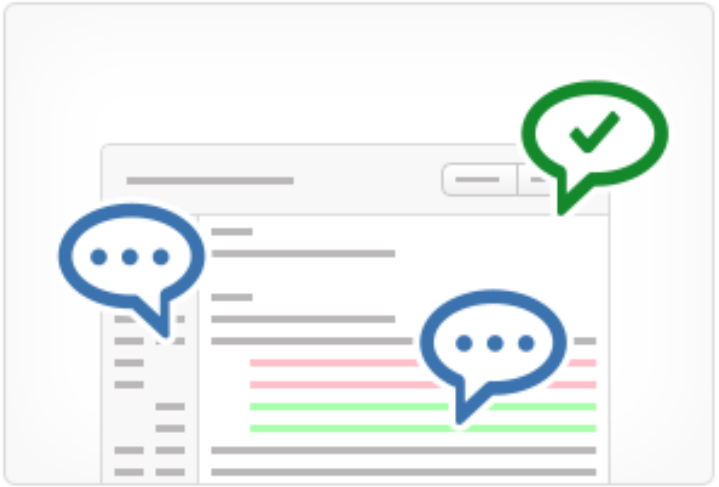
Collaborate with pull requests

Pull requests let you tell others about changes.

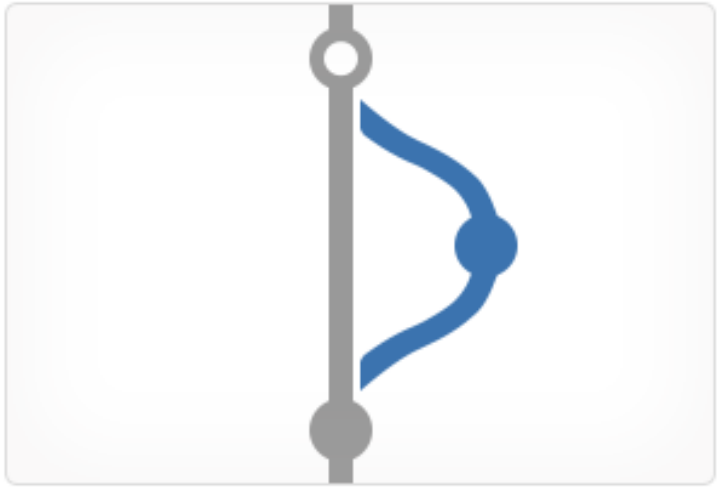
Review and merge your code in a single collaborative process.

Be sure to provide good feedback and protect branches with policies.

Branch
Develop features on a branch and create a pull request to get changes reviewed.

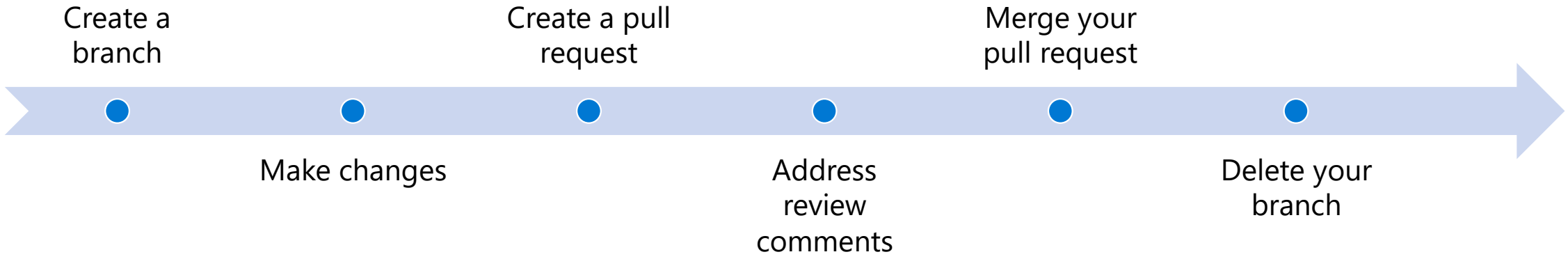


Discuss
Discuss and approve code changes related to the pull request.

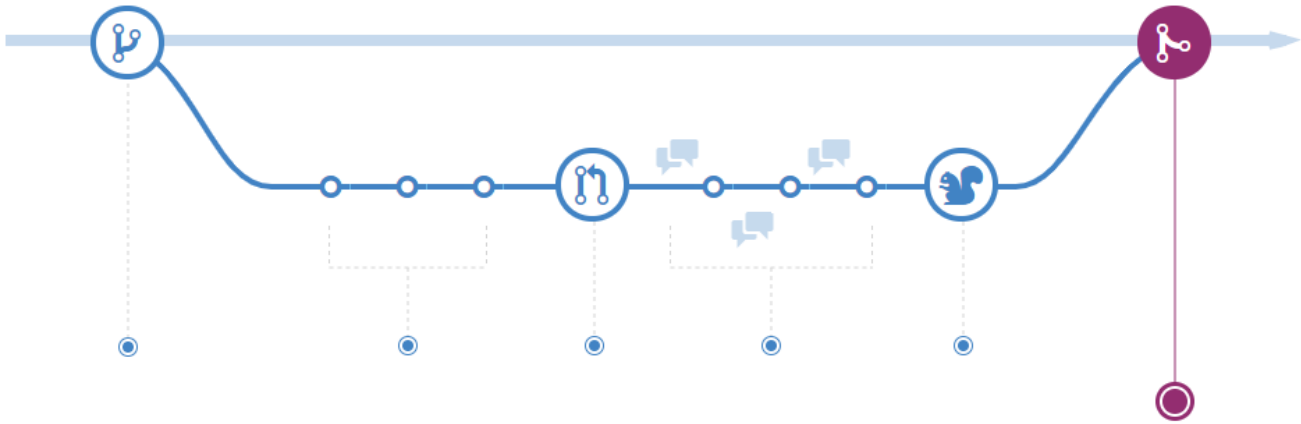


Merge
Merge the branch with the click of a button.

Explore GitHub flow



GitHub Flow is a lightweight, branch-based workflow. The GitHub flow is useful for everyone, not just developers



Demo

GitHub In Action

GitHub Codespaces and Team Collaboration

What is Microsoft Teams?

Microsoft Teams is the hub for teamwork.

Key features of Teams

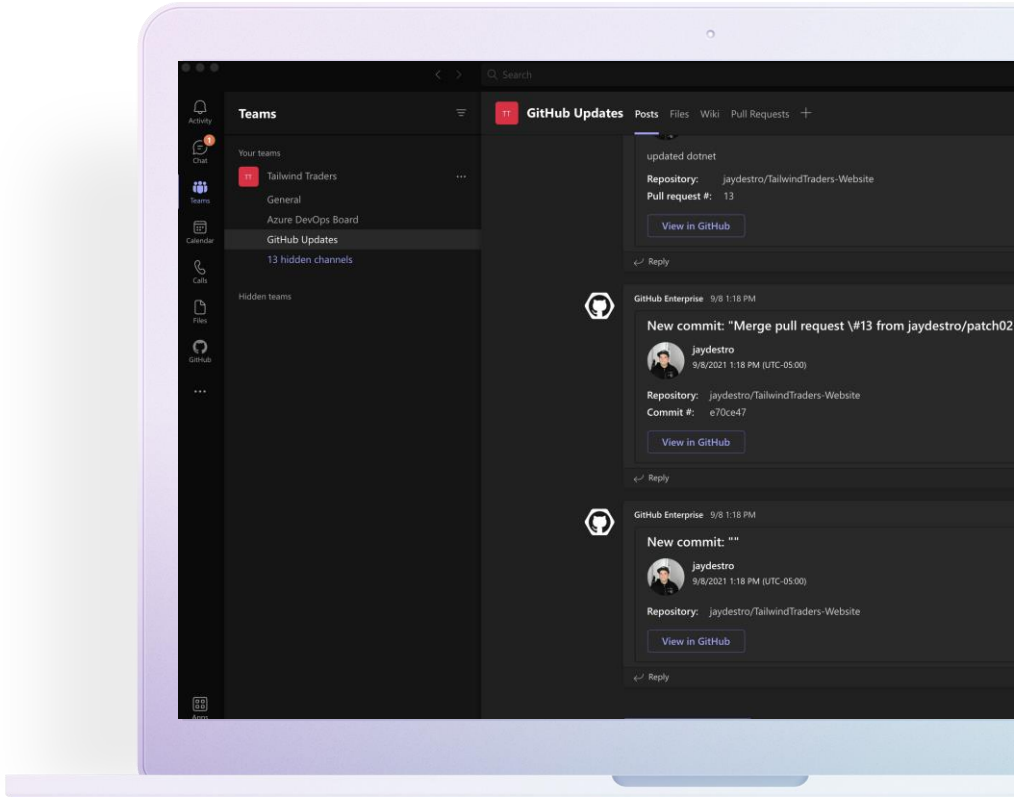
Chat from anywhere

Meet from anywhere

Call from anywhere

Collaborate from anywhere

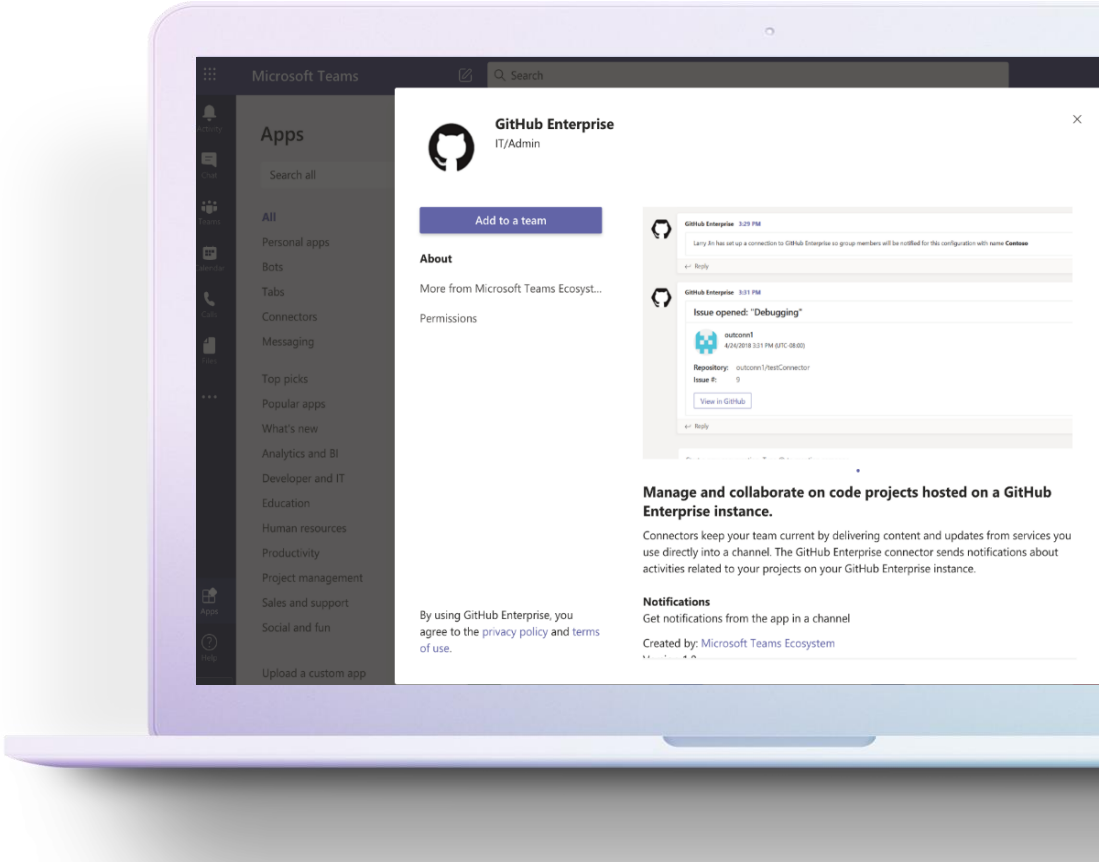
Achieve more, faster



Learn about Microsoft Teams

Microsoft Teams integration via GitHub Enterprise app

Uses a connector to send notifications directly into a specific Microsoft Teams channel.



Demo

Collaboration with Teams

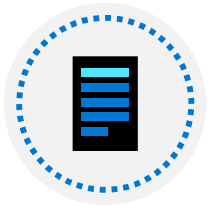
What is GitHub Codespaces?

- Codespaces is a cloud-based development environment that GitHub hosts.
  - Syntax highlighting.
  - Autocomplete.
  - Integrated debugging.
  - Direct Git integration.
- GitHub Codespaces addresses several issues from which developers regularly suffer.
- You can do all your work in Codespaces within a browser.

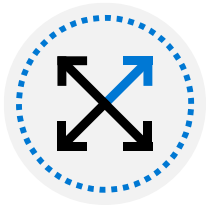
Develop online with GitHub Codespaces

Avoids issues with old hardware/software

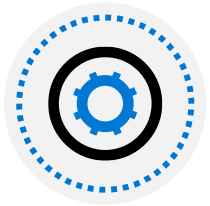
Based on Visual Studio Code



Highly portable



Work from PCs, tablets, Chromebooks

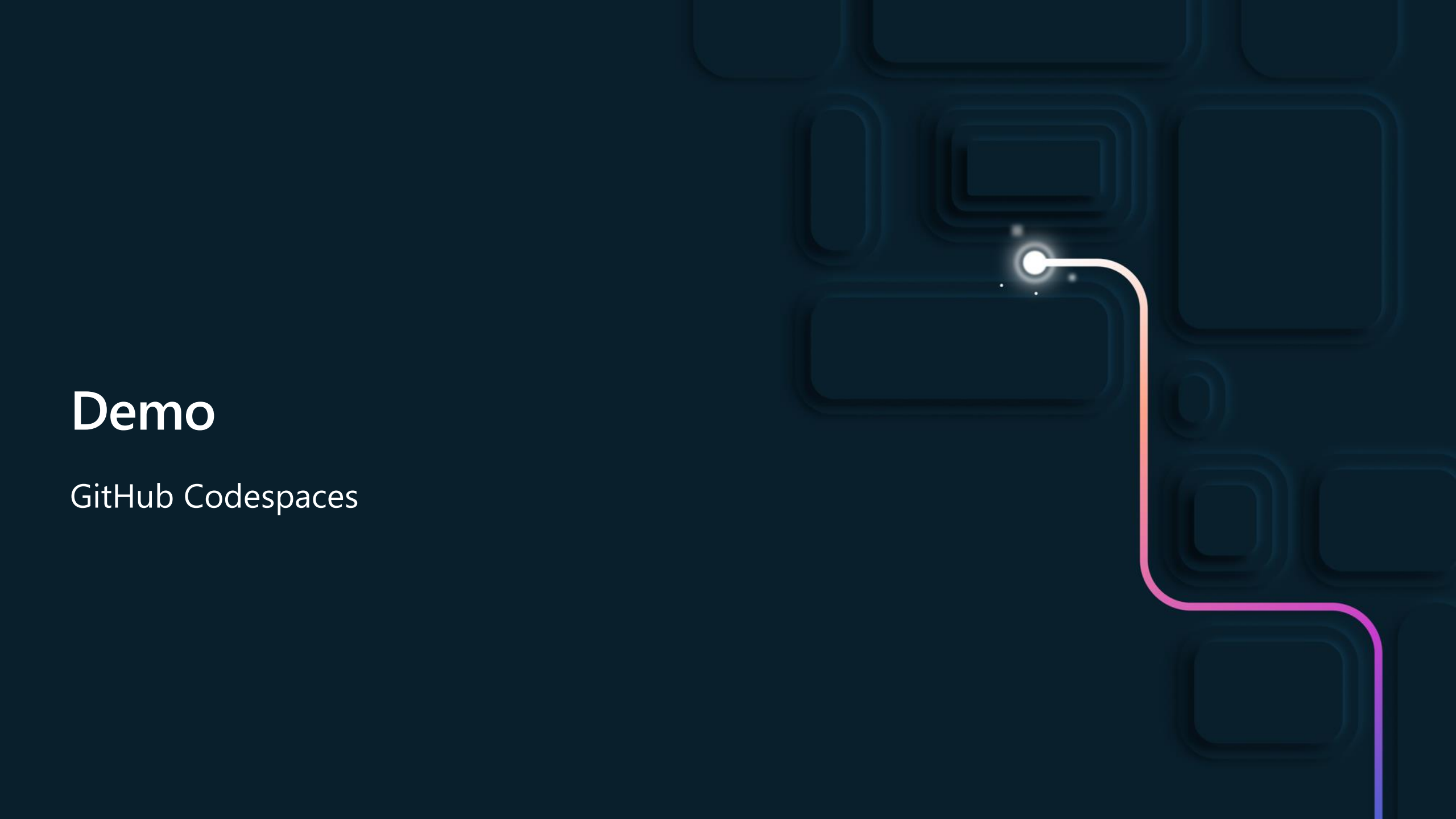


Protect against proliferation of intellectual property

Connect to Codespaces from Visual Studio Code

Demo

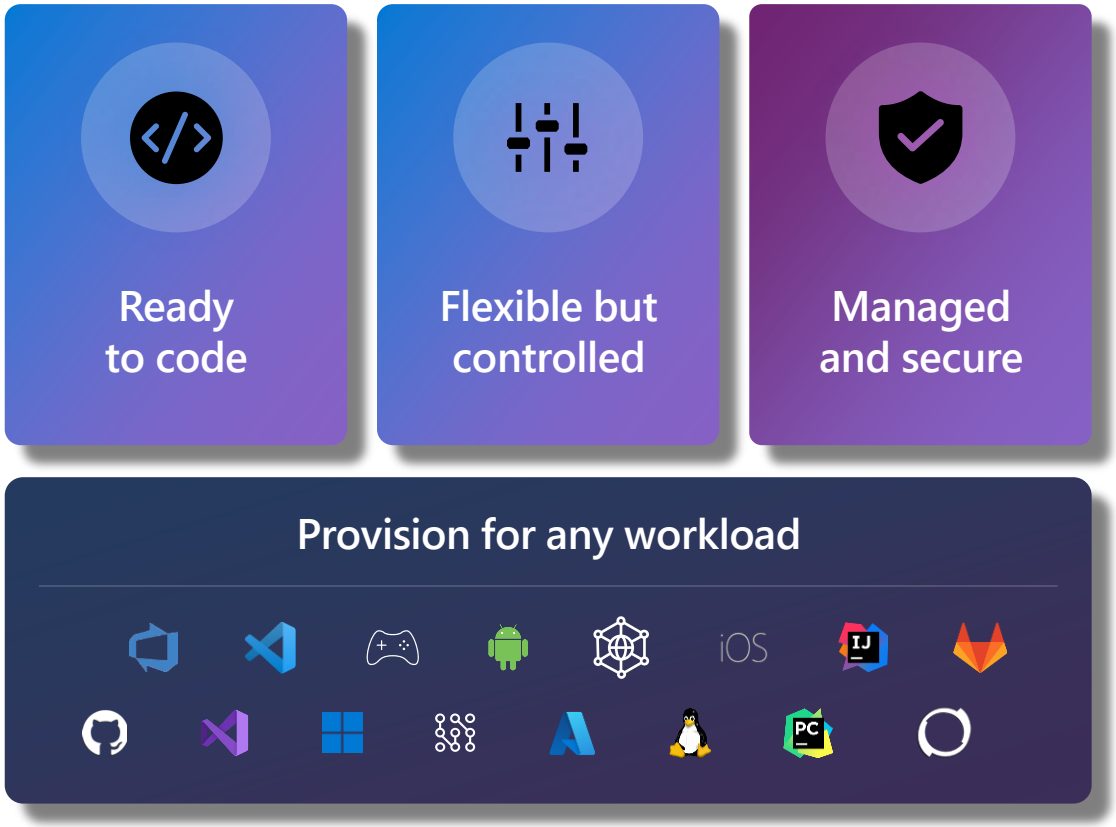
GitHub Codespaces



Introduction to Microsoft Dev Box and its role in DevOps

Self-service development with Microsoft Dev Box

Ready-to-code, secure dev workstations for a hybrid team



Microsoft Dev Box capabilities

- Self-service Dev Box lifecycle management
- Ready-to-code with task-focused images
- Dev Box hibernation and easy restart

Any tool and any workload

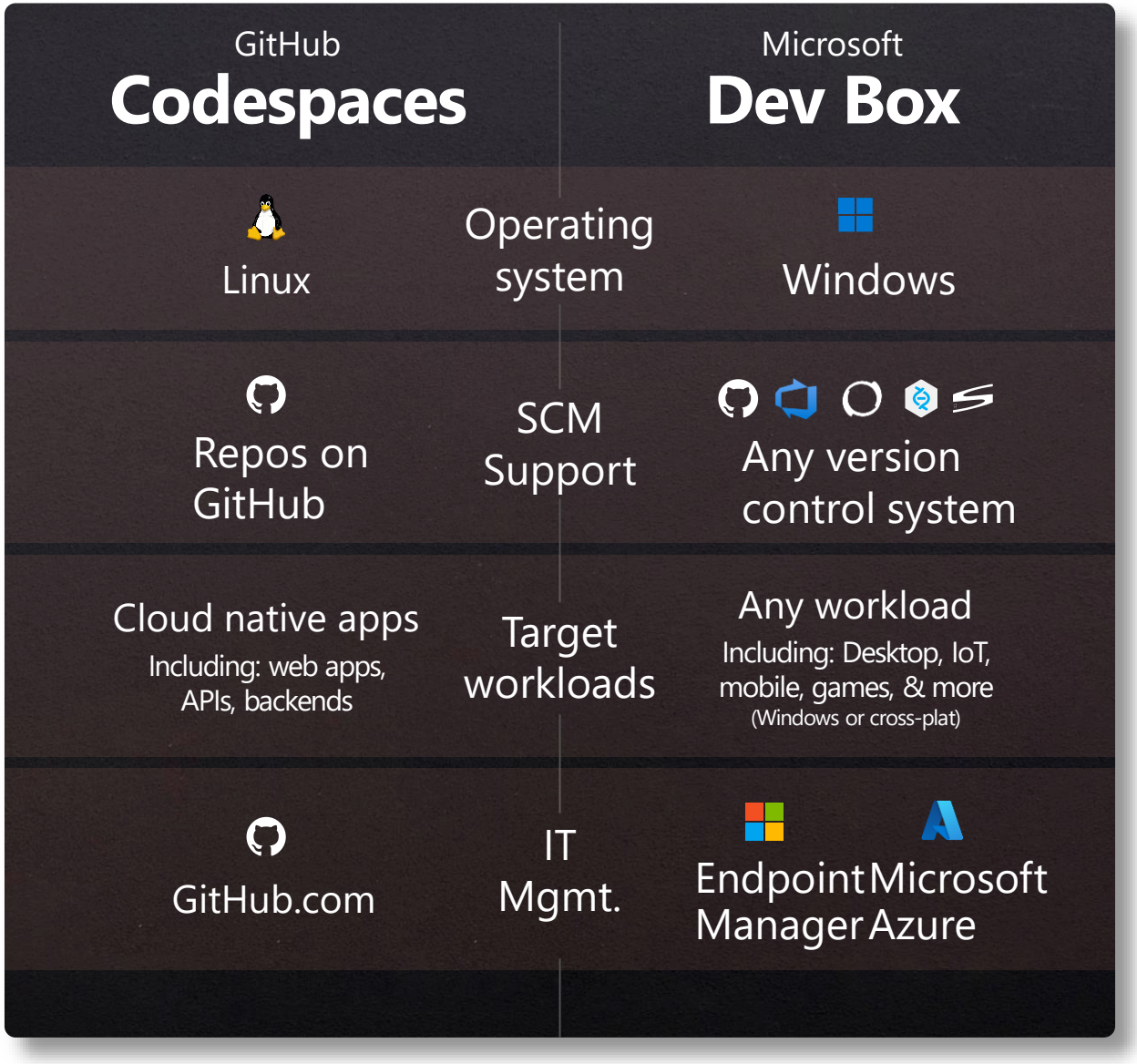
- Accessible on any OS or browser
- Works with the latest Windows versions
- Secured and centrally managed

Deploy on any device

Own your workstation

- Dedicated compute to match project demands
- Deploy any IDE, SDK tools that run on Windows
- Develop for desktop, mobile, web, and more
- WSL and nested virtualization support
- Day-to-day development
- Separate Dev Boxes for different projects
- Proof of Concepts
- Maintaining legacy applications

Complimentary services: Codespaces + Microsoft Dev Box

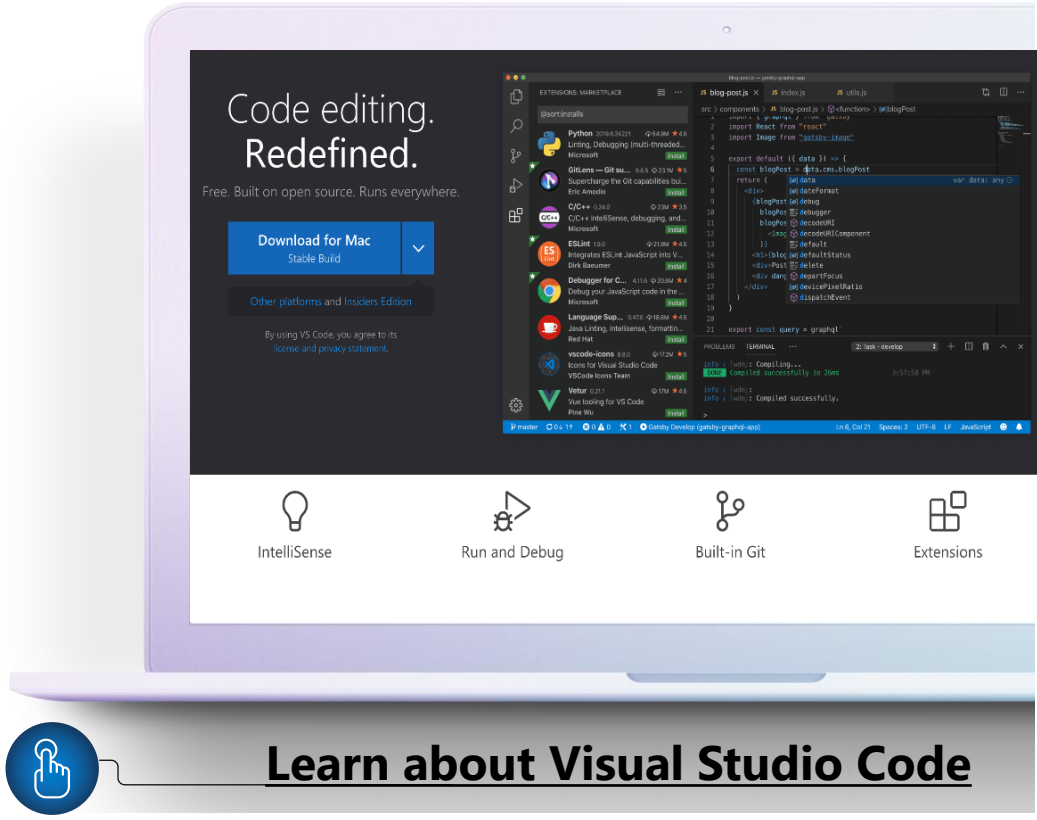


Extending DevOps with Visual Studio Code

What is Visual Studio Code?

Visual Studio Code is a lightweight and powerful source code editor.

- Run anywhere (Mac, Win, Lin)
- Git commands built-in
- Extensible and customizable
- IntelliSense syntax highlights
- Easily debug code
- Open Source
- Free!



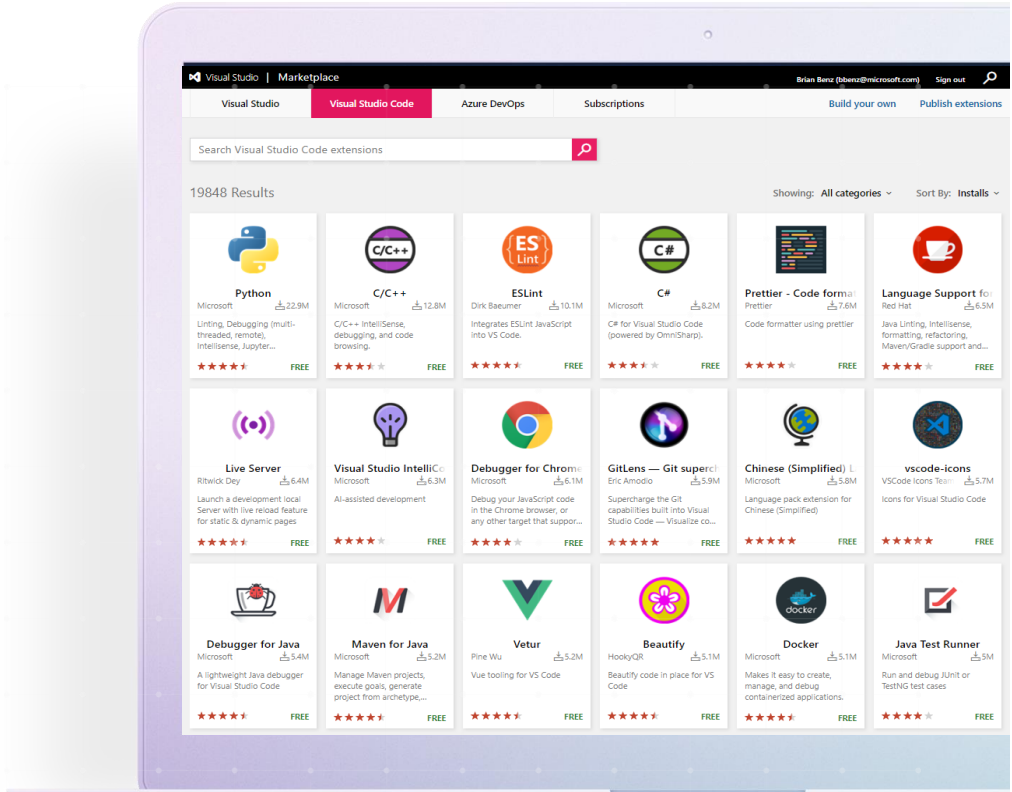
Learn about Visual Studio Code

Visual Studio Code Marketplace

Thousands of extensions!

Microsoft and Third-party

- Language support
- Tooling support
- Connectivity
- Deployment
- Multi-cloud

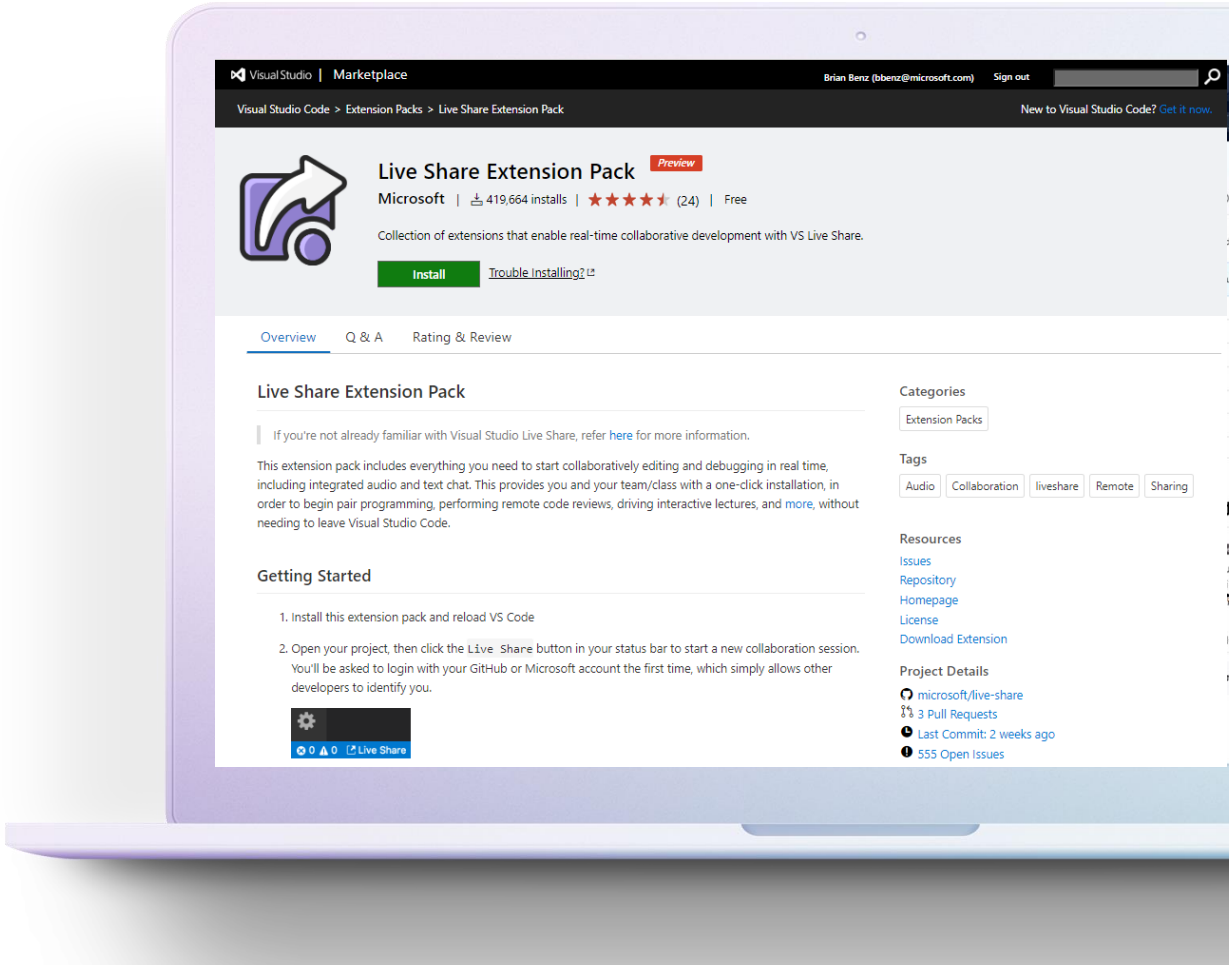


Learn about Visual Studio Code Marketplace

Extension example – Live Share extension pack

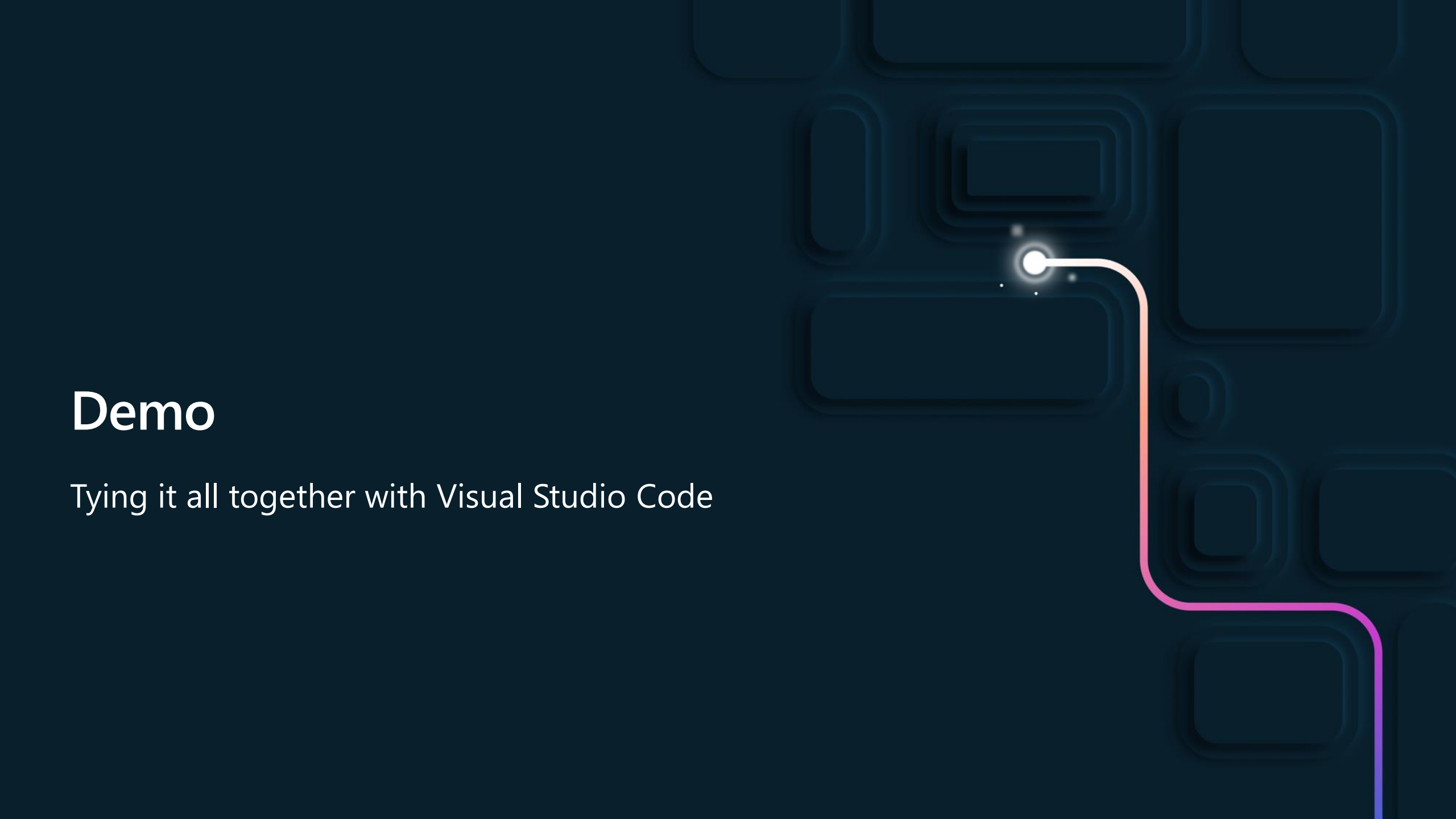
Share your code and collaborate

- Share terminals and servers
- Edit
- Debug
- Audio calls
- Chat



Demo

Tying it all together with Visual Studio Code



DevOps and Tailwind Traders

Recap

TAILWIND TRADERS

Needed Solving

- Rapidly growing
- Lack of collaboration
- Lack of shared tooling

The Solution

- Source control via GitHub
- Microsoft Teams collaboration hub
- Visual Studio Code for shared tooling
- Solid foundation for a DevOps strategy

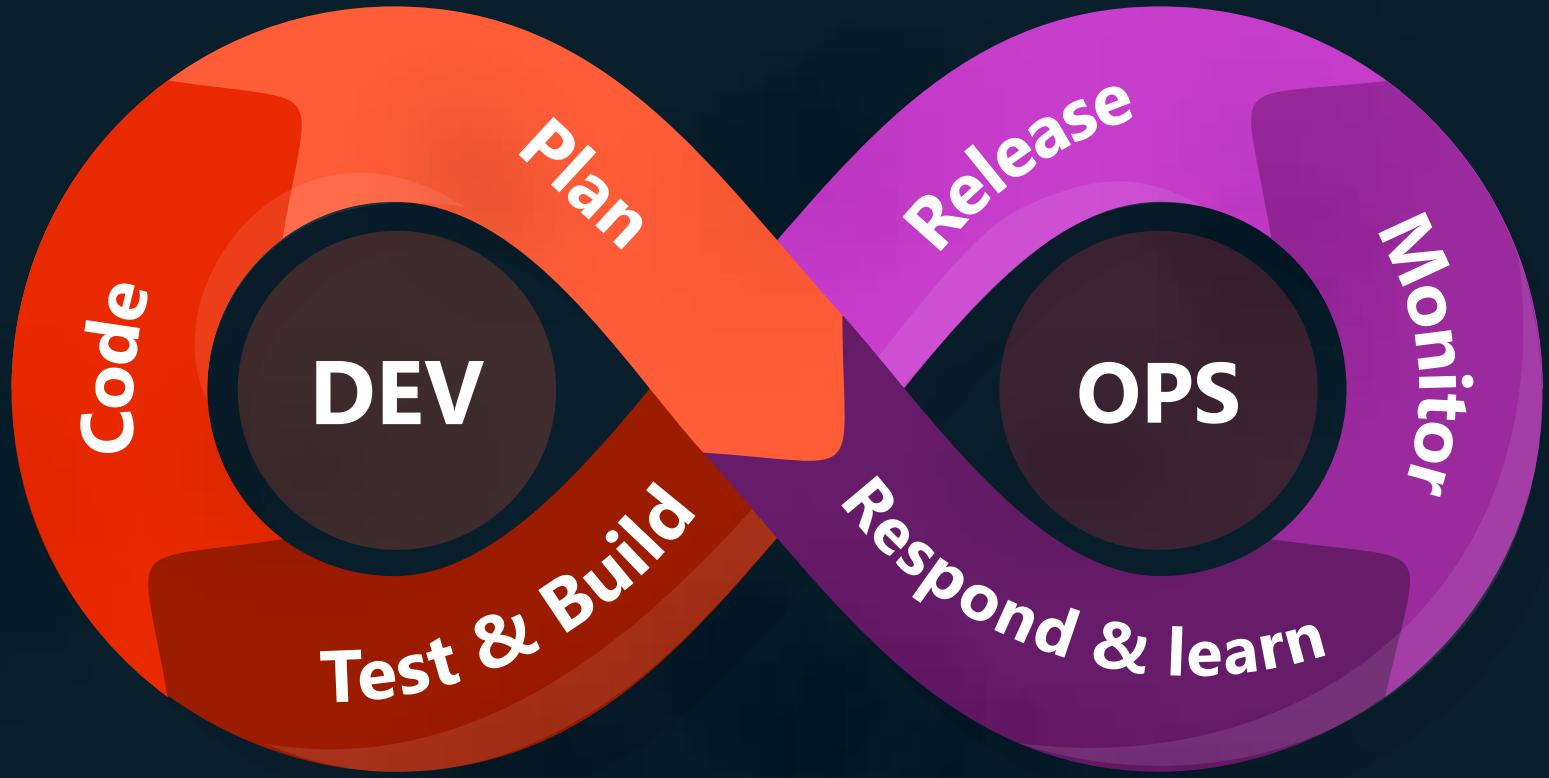
Microsoft

Managing the Flow of Work

DevOps Learning Path

- Getting Started with DevOps
- **Managing the Flow of Work**
- Shift security “left” in your CI/CD process
- Delivering changes to cloud
- Performance Monitoring and maintenance

Tailwind Traders all in on DevSecOps




"
DevSecOps is the culture of integrating and automating preventive, detective, and responsive security controls into the pipeline.
"

Why is DevSecOps so Important?

Shift Left Security

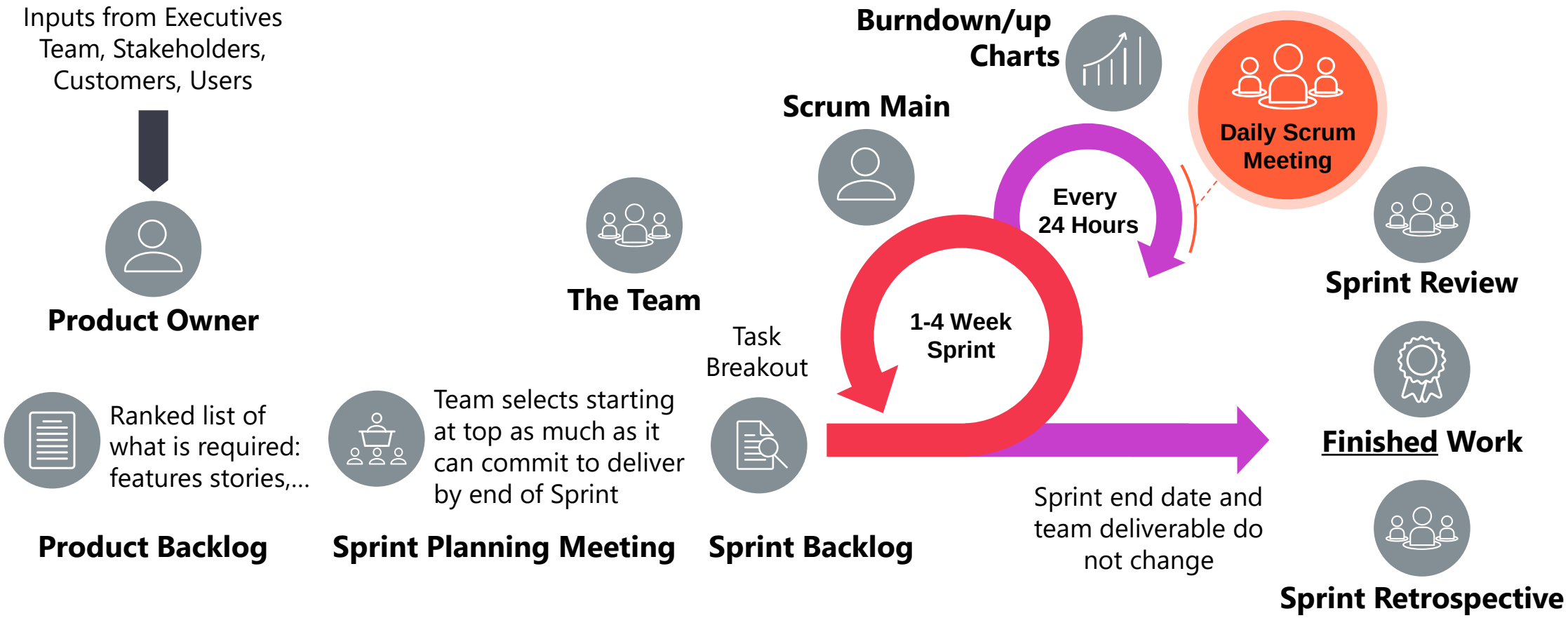
Your competition is already doing this

Increase velocity

Reduced downtime

Reduced human error

The Agile: Scrum Framework at a glance



General principles

Product is built incrementally

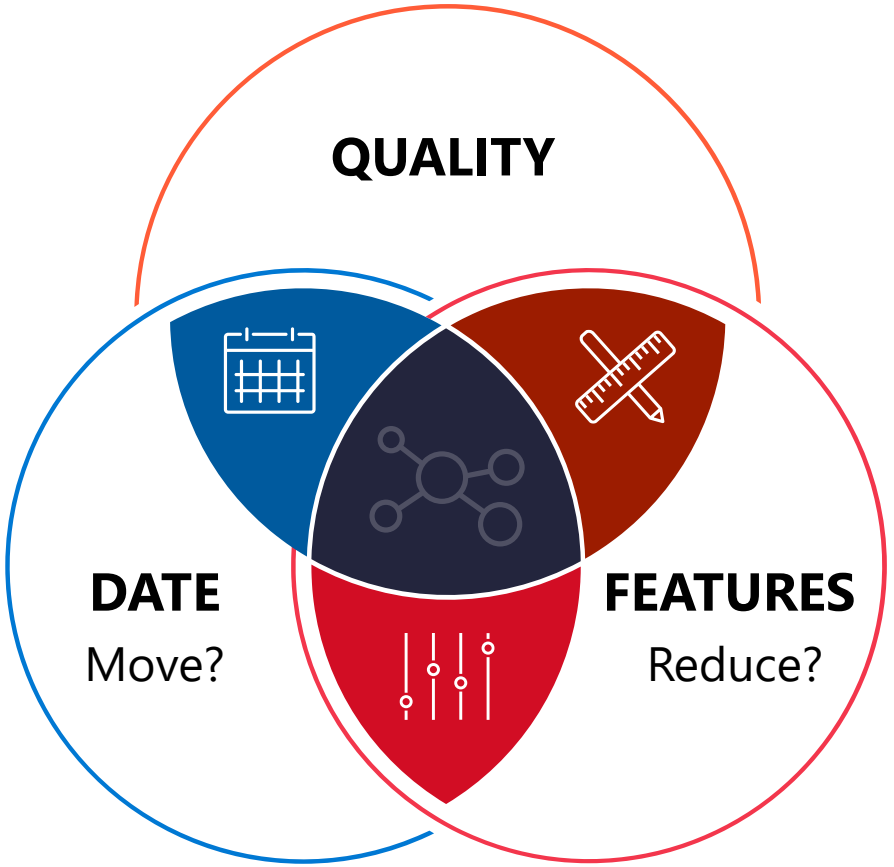
Frequent inspection and adaption (course correction)

Transparency (Product and Sprint backlogs are public)

Product Owner, Development Team, Scrum Main

Scrum Teams are self-organizing and cross-functional

Quality is non-negotiable

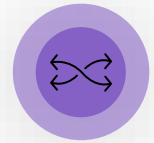


Estimates

Never accept an estimate over 4 hours

More accurate

Enables parallel development



Confirms alignment with DoD

Never start from a date

The rules apply to everyone

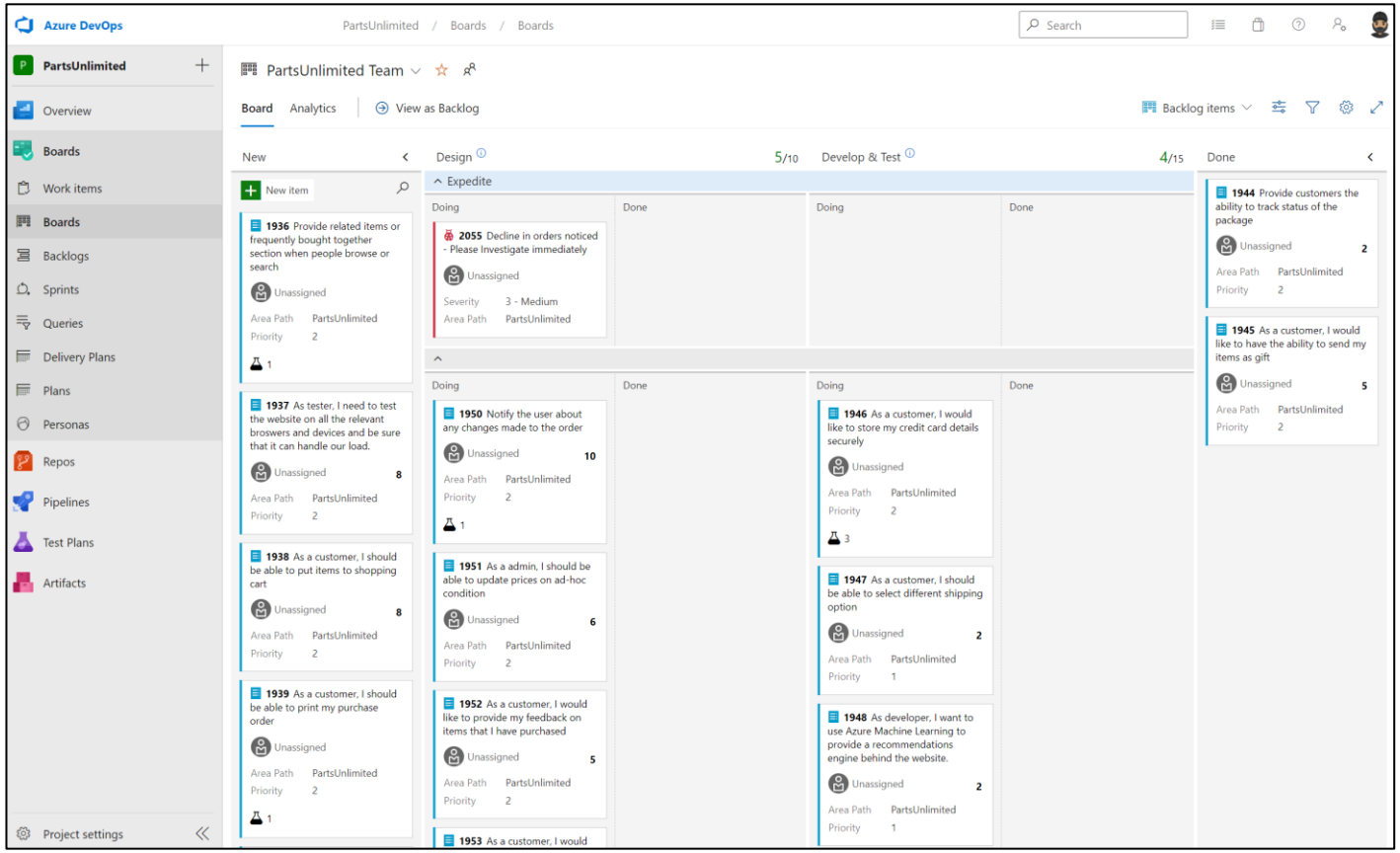
Even the CEO must obey the rules

No one is above the law

Work with Azure Boards

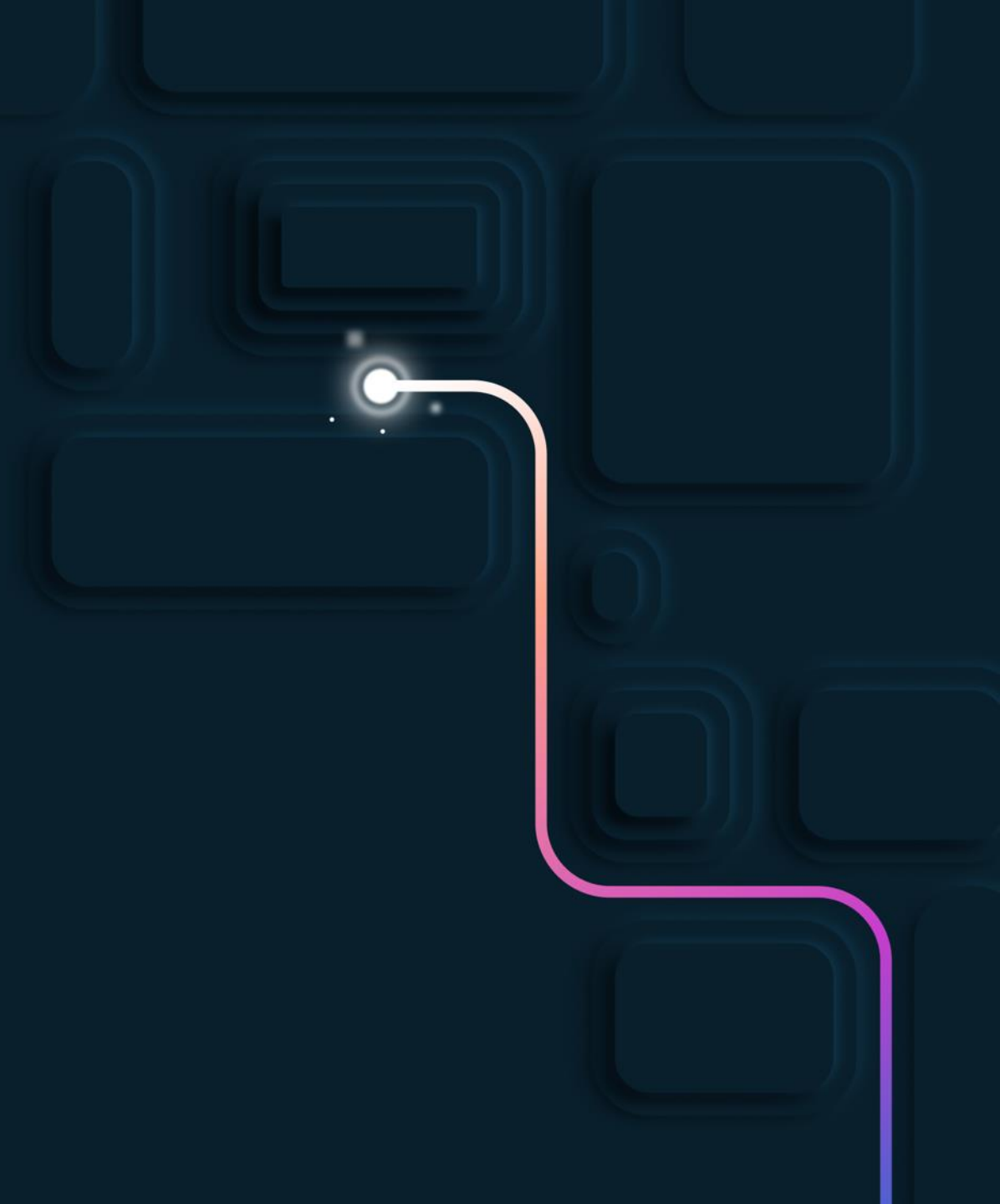
Agile, Scrum, and Kanban processes by default.

Track work, issues, and code defects associated with your project.



Demo

Tracking Work using Azure Boards

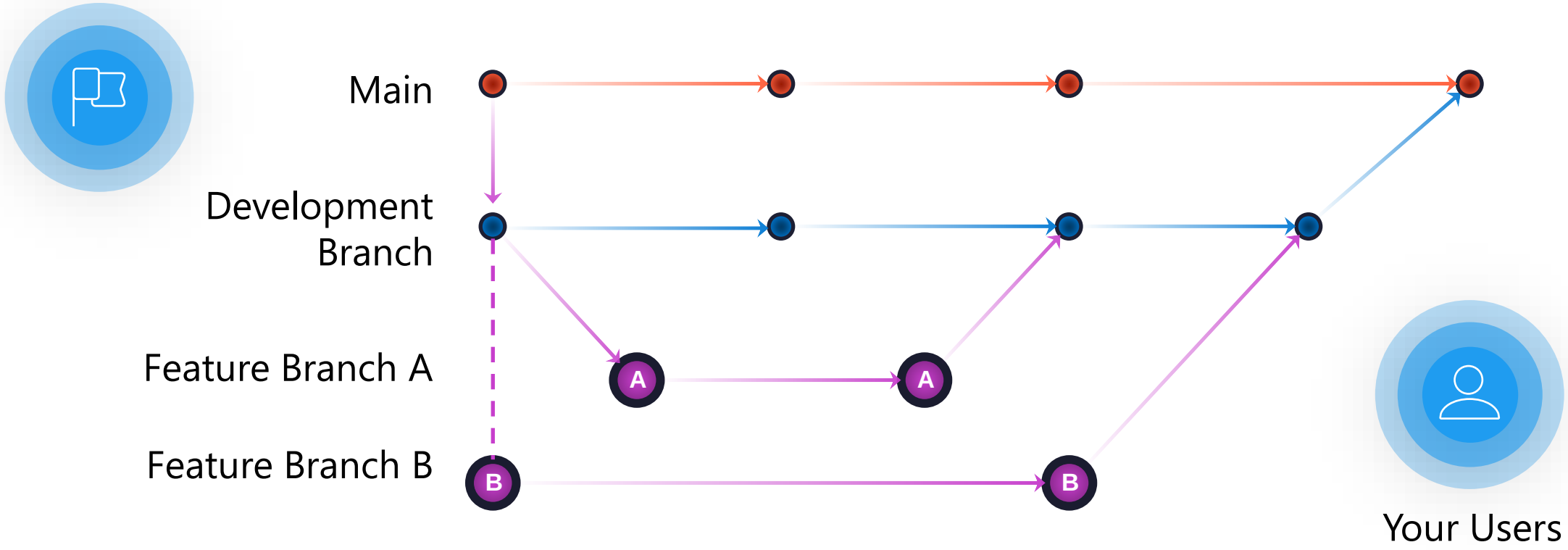


Branching Strategy

Choosing the right branching scheme is critical to success

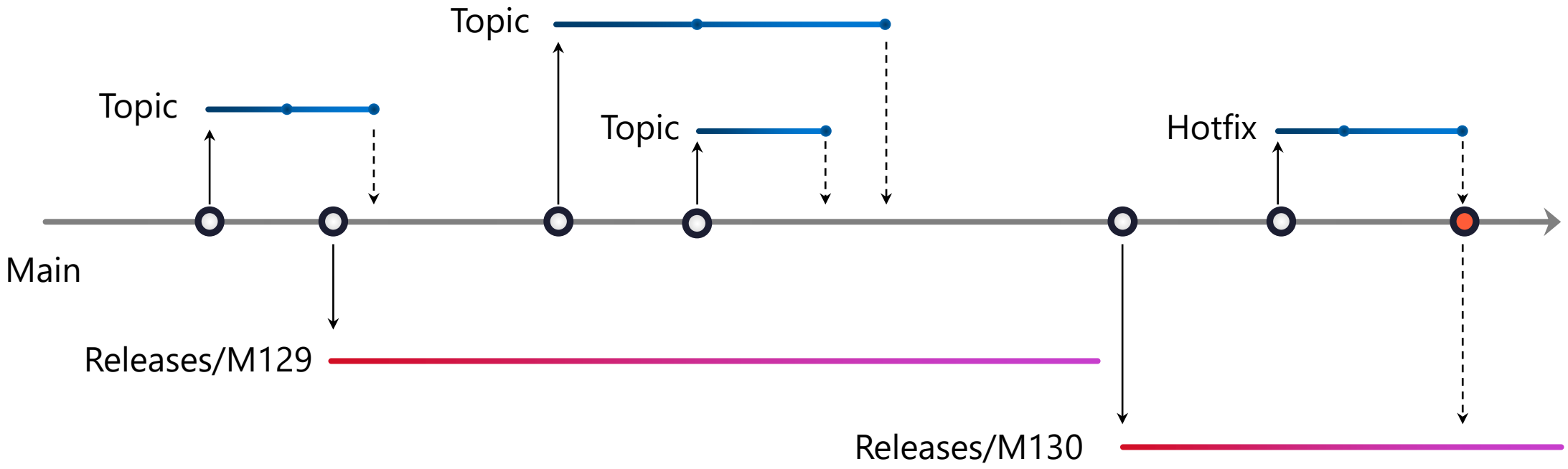
Traditional branching strategy

Feature Branching without flags



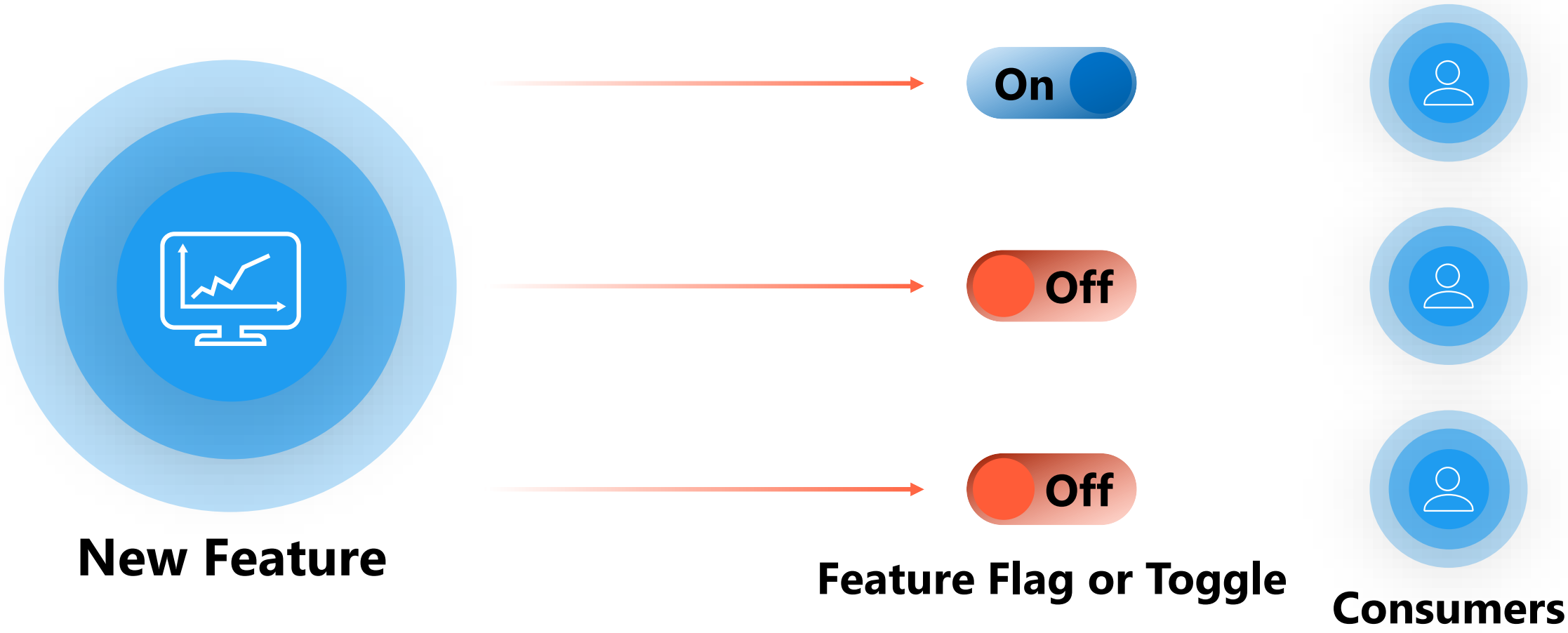
Trunk based development

Using trunk-based development to avoid merge debt



How can that work?

Feature flags



Glorified If statement

Bob

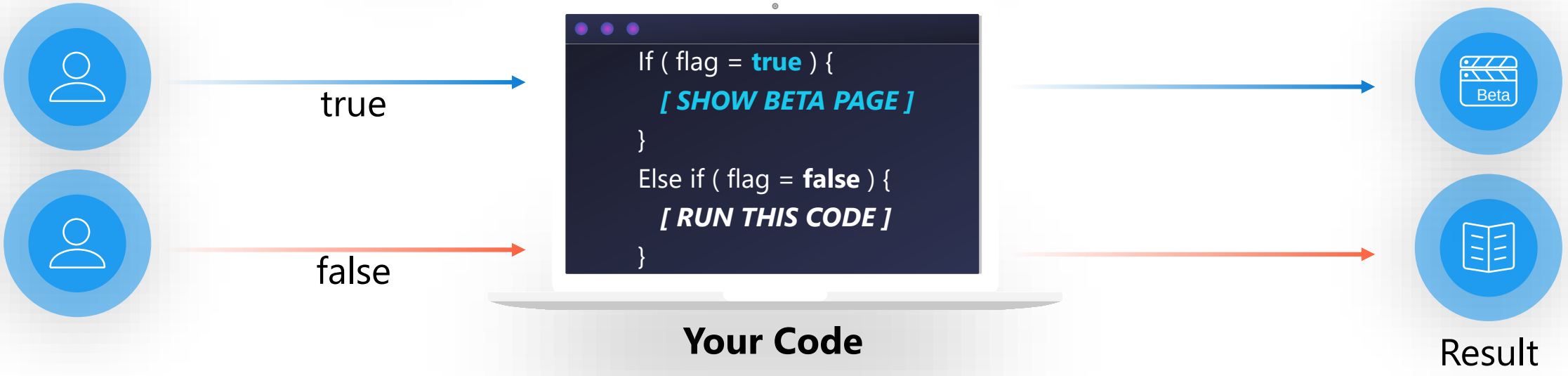
{
Key: bob@example.com",
name: "Bob Smith",
group: "**beta**"
}

Sarah

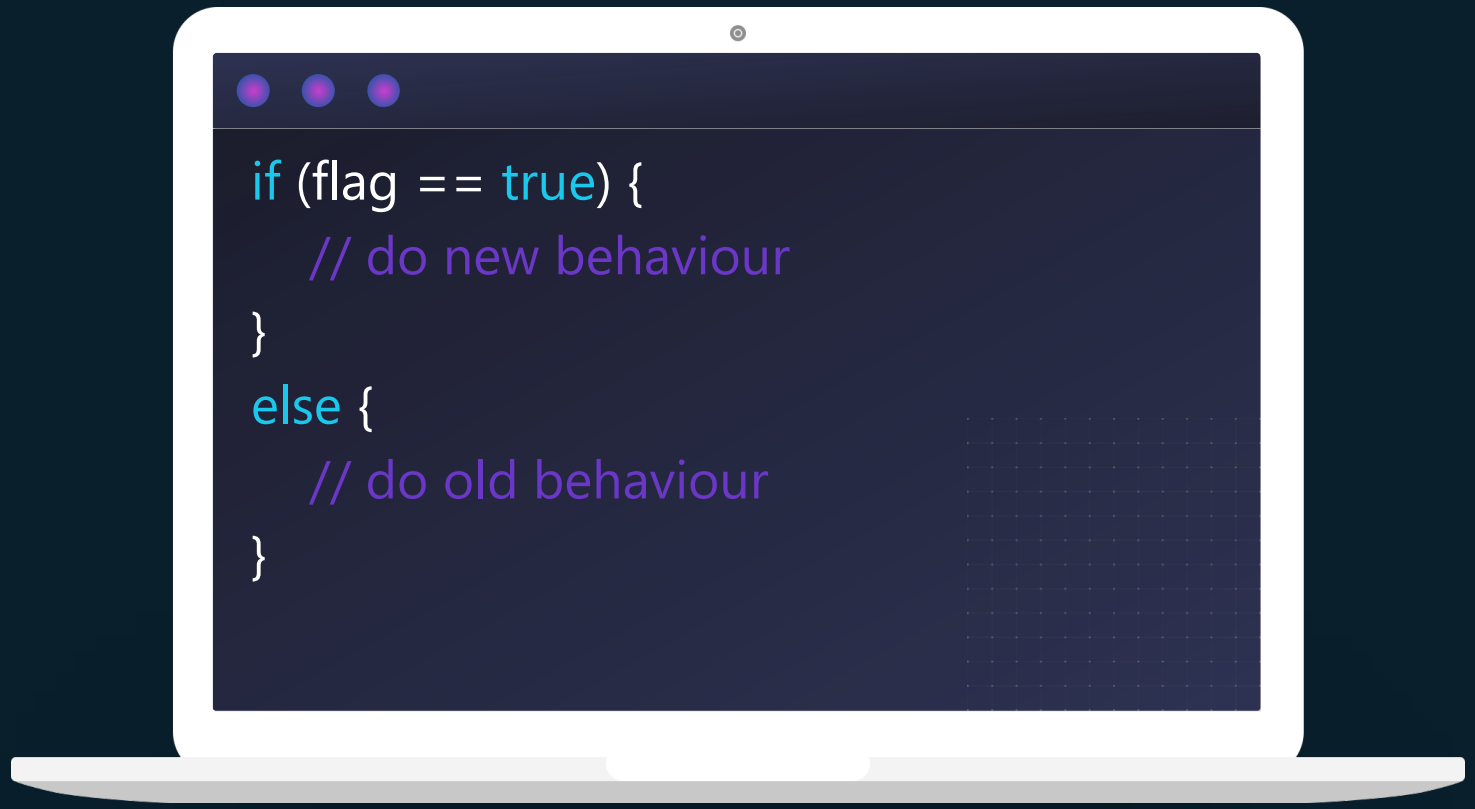
{
Key:
sarah@example.com",
name: "Sarah Jones",
group: "normal"
}

"Beta Page"

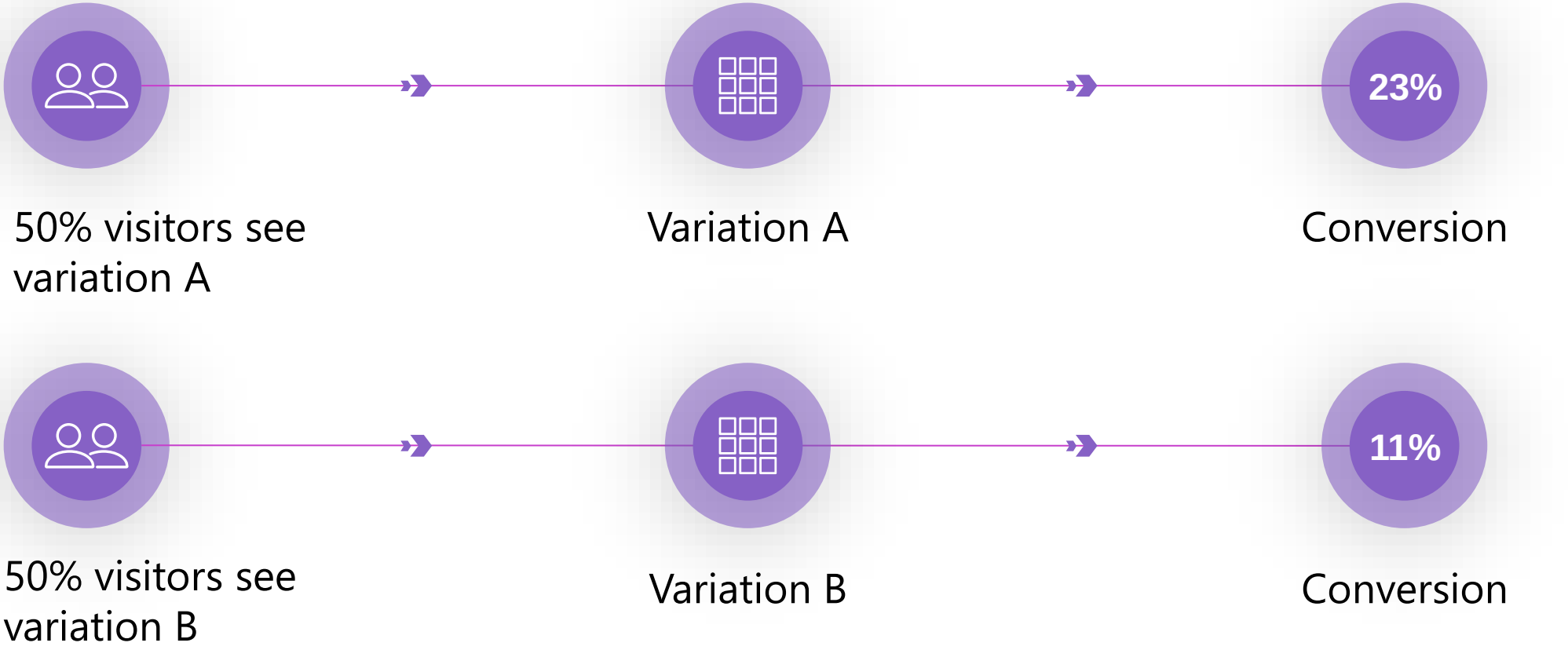
If group is **beta** return **true**
... if not, return **false**



No really… it’s an if statement

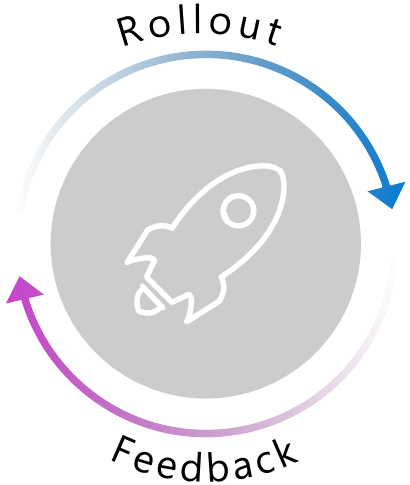


A/B experiments



Safe deployment

New Feature



Soft Launch

Incremental Rollout

Set switch to off. Done.

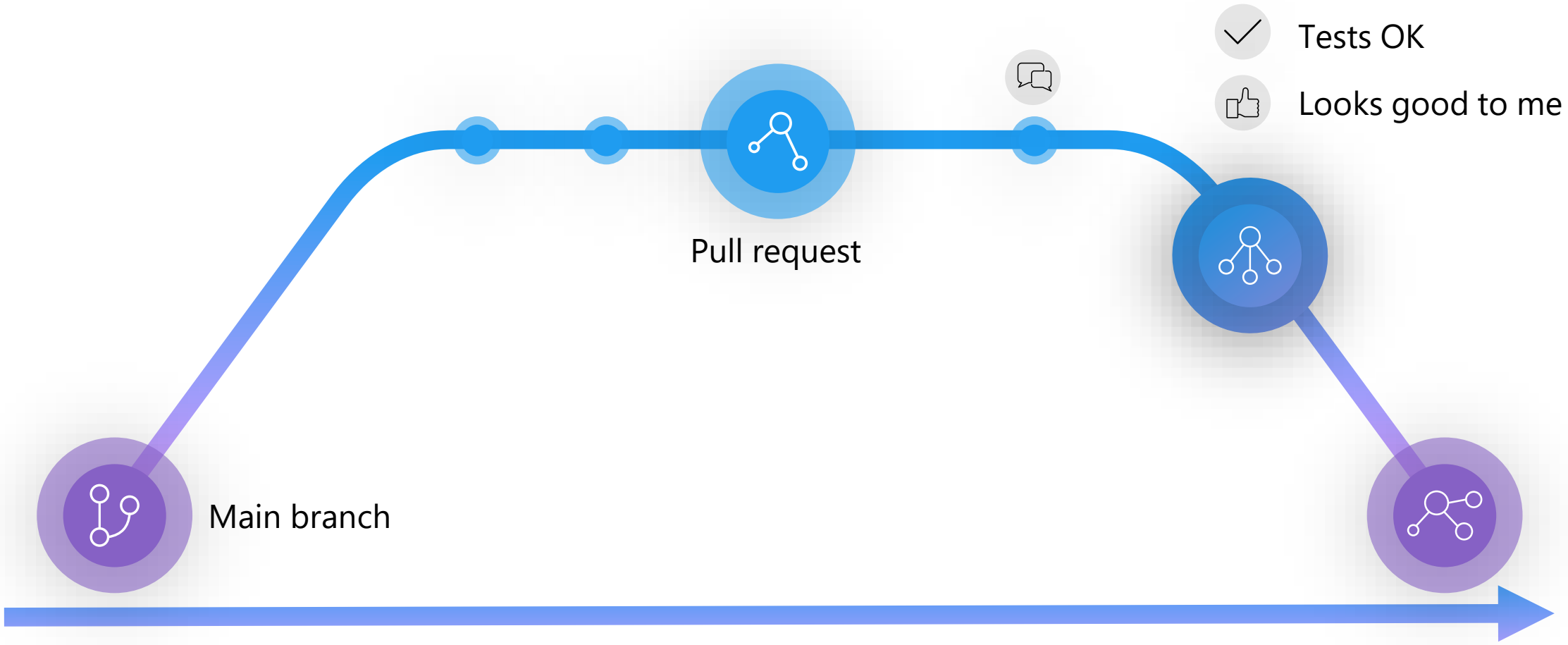
Rollback

Done

Demo

Feature Flags

Maintaining quality w/pull requests



Automation to the rescue!

Infrastructure as Code

Bicep

What is Bicep?

Azure Bicep is the next revision of **ARM templates** designed to solve some of the issues developers were facing when deploying their resources to Azure.

Note: Beware that when converting ARM templates to Bicep, there might be issues since it's still a work in progress.

```
param storageName string =
'stg${uniqueString(resourceGroup().id)}'
param location string = resourceGroup().location

resource storageaccount
'Microsoft.Storage/storageAccounts@2021-02-01' = {
    name: 'name'
    location: location
    kind: 'StorageV2'
    sku: {
        name: 'Premium_LRS'
    }
}
```

Understand Bicep file structure and syntax

Azure Bicep comes with its own syntax, however, it's easy to understand and follow:

- Scope, Parameters, Variables, Resources, Modules, Outputs.

Other features: loops, conditional deployment, multiline strings, referencing an existing cloud resource, and many more.

```
@minLength(3)
@maxLength(11)
param storagePrefix string

param storageSKU string = 'Standard_LRS'
param location string = resourceGroup().location

var uniqueStorageName = '${storagePrefix}${uniqueString(resourceGroup().id)}'

resource stg 'Microsoft.Storage/storageAccounts@2019-04-01' = {
    name: uniqueStorageName
    location: location
    sku: {
        name: storageSKU
    }
    kind: 'StorageV2'
    properties: {
        supportsHttpsTrafficOnly: true
    }

    resource service 'fileServices' = {
        name: 'default'

        resource share 'shares' = {
            name: 'exampleshare'
        }
    }
}

module webModule './webApp.bicep' = {
    name: 'webDeploy'
    params: {
        skuName: 'S1'
        location: location
    }
}

output storageEndpoint object = stg.properties.primaryEndpoints
```

Demo

Deploy a Bicep file from GitHub workflows

Tailwind Traders is now in a good state

Let’s Recap

TAILWIND TRADERS

Needed Solving

- Managing Work
- Managing Source Control Changes
- Automation to help with processes

The Solution

- Scrum and Azure Boards
- Trunk Based Development
- Feature Flags
- GitHub Actions

Microsoft

Shift Security “Left” in your CI/CD Process

DevOps Learning Path

- Getting Started with DevOps
- Managing the Flow of Work
- **Shift security “left” in your CI/CD process**
- Delivering changes to cloud
- Performance Monitoring and maintenance

Goals for this Session

Security

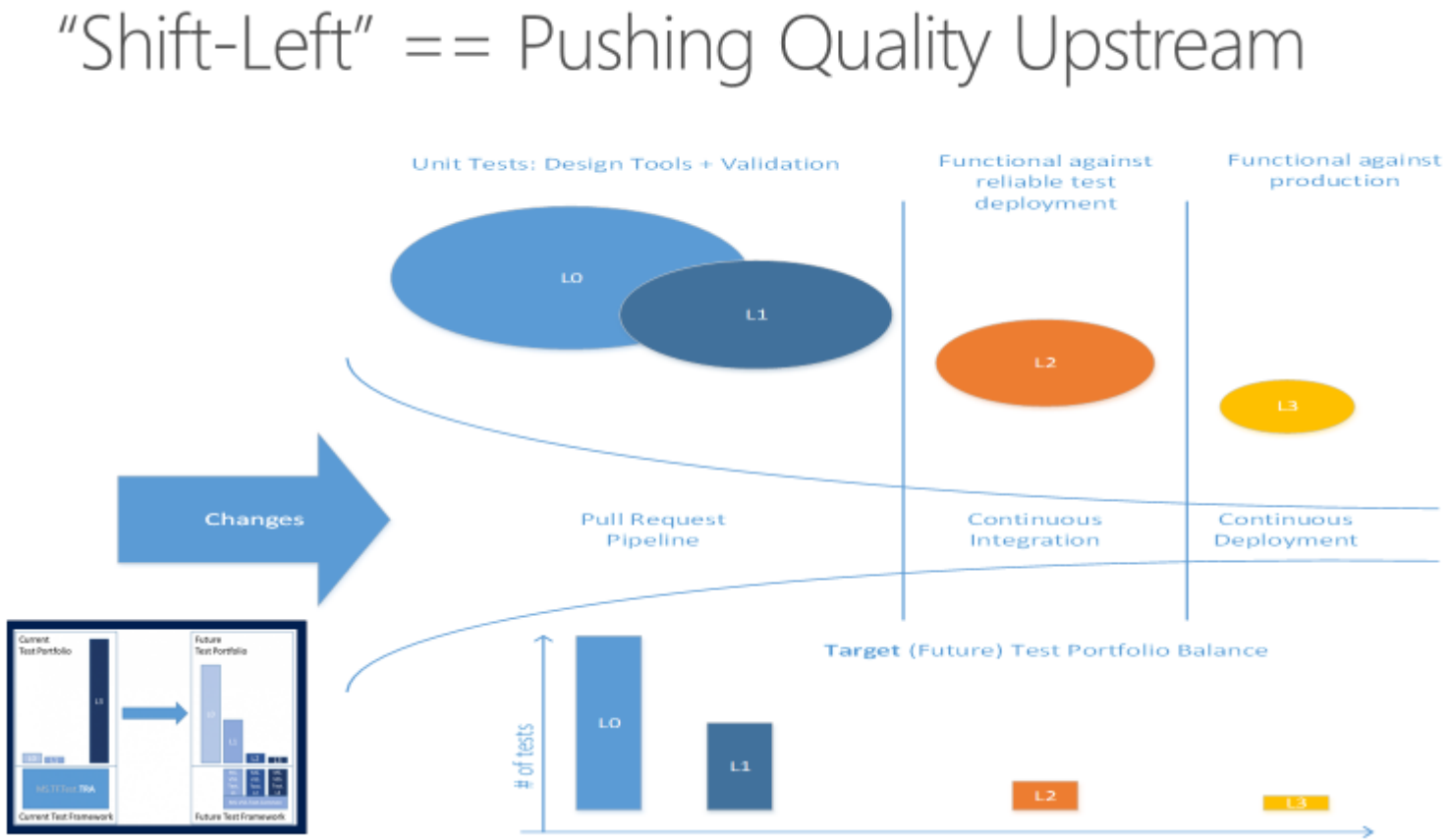
Apply Security to Containers

Build Quality and Gain Confidence

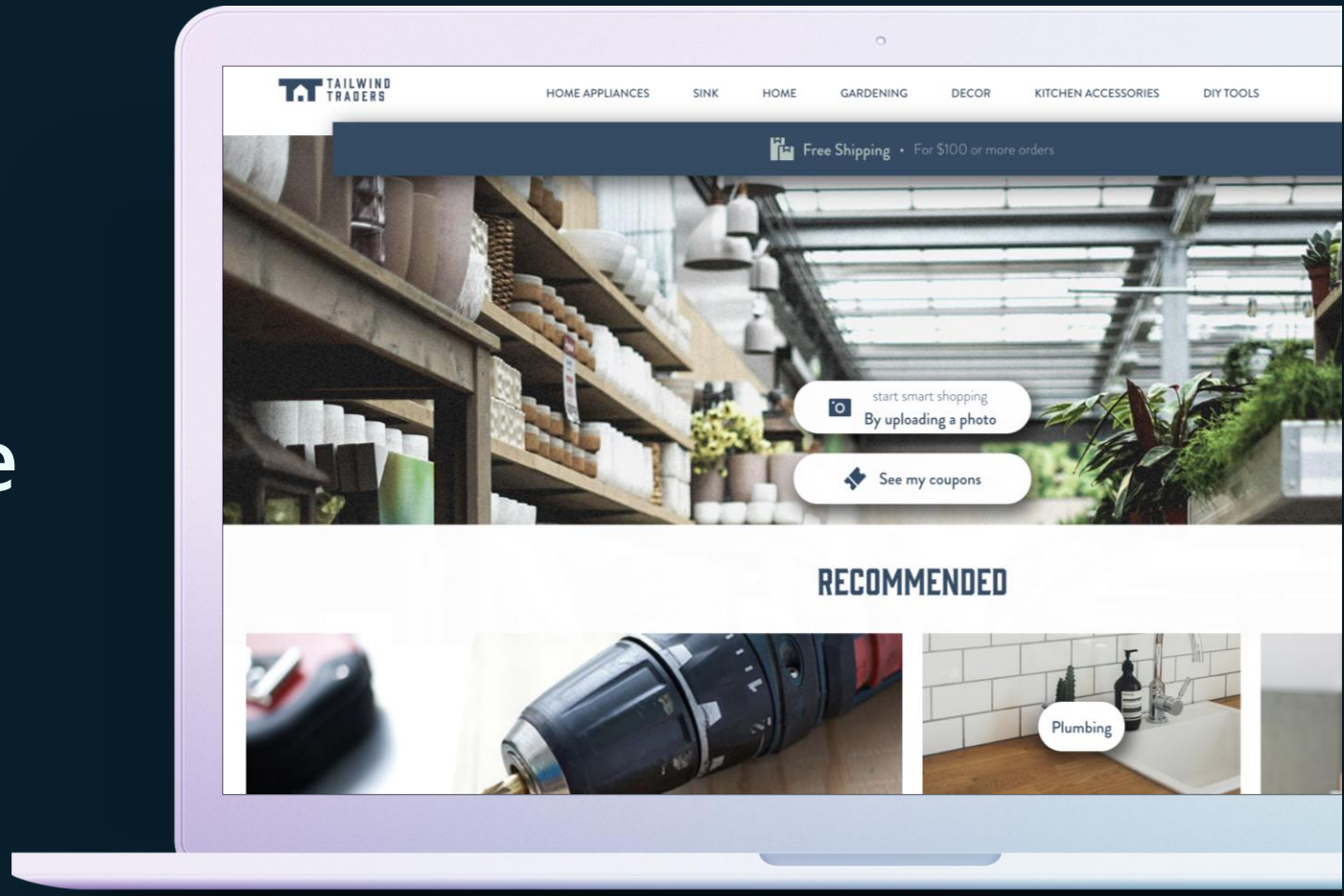
Understand Shift-left

The goal for shifting left is to move quality upstream by performing tests early in the pipeline.

Combine test and process improvements to reduce the time it takes for tests.



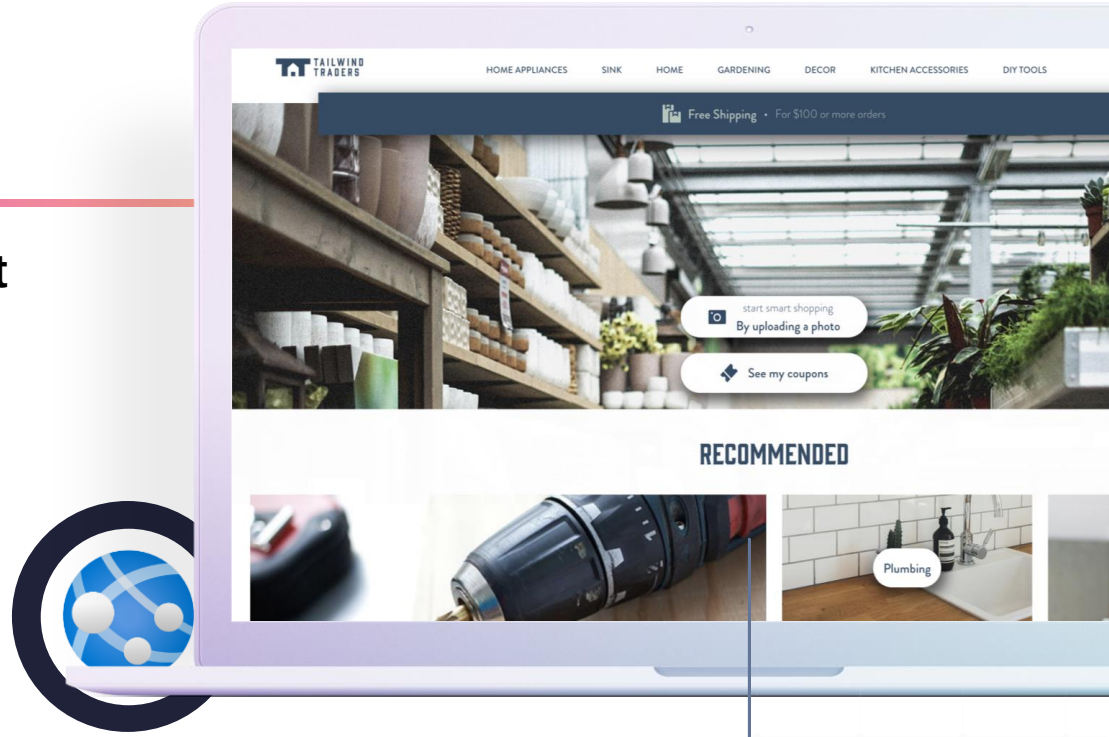
Tailwind Traders Website



Tailwind Traders Website

ASP.NET Core + React

Docker container on
Azure App Service



Azure SQL Database

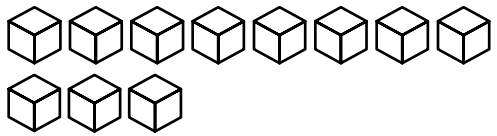
Azure Cosmos DB

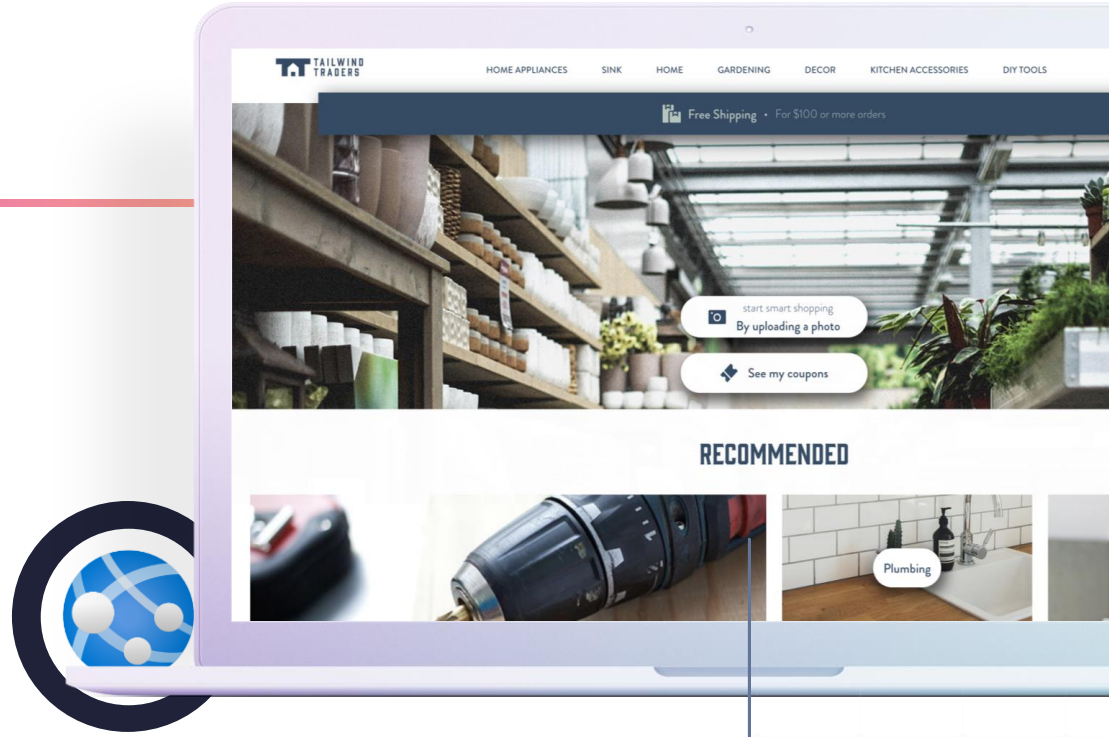
What Tailwind Traders Needs

Tailwind Traders Website

ASP.NET Core + React

Microservices with backend APIs and Web Frontend



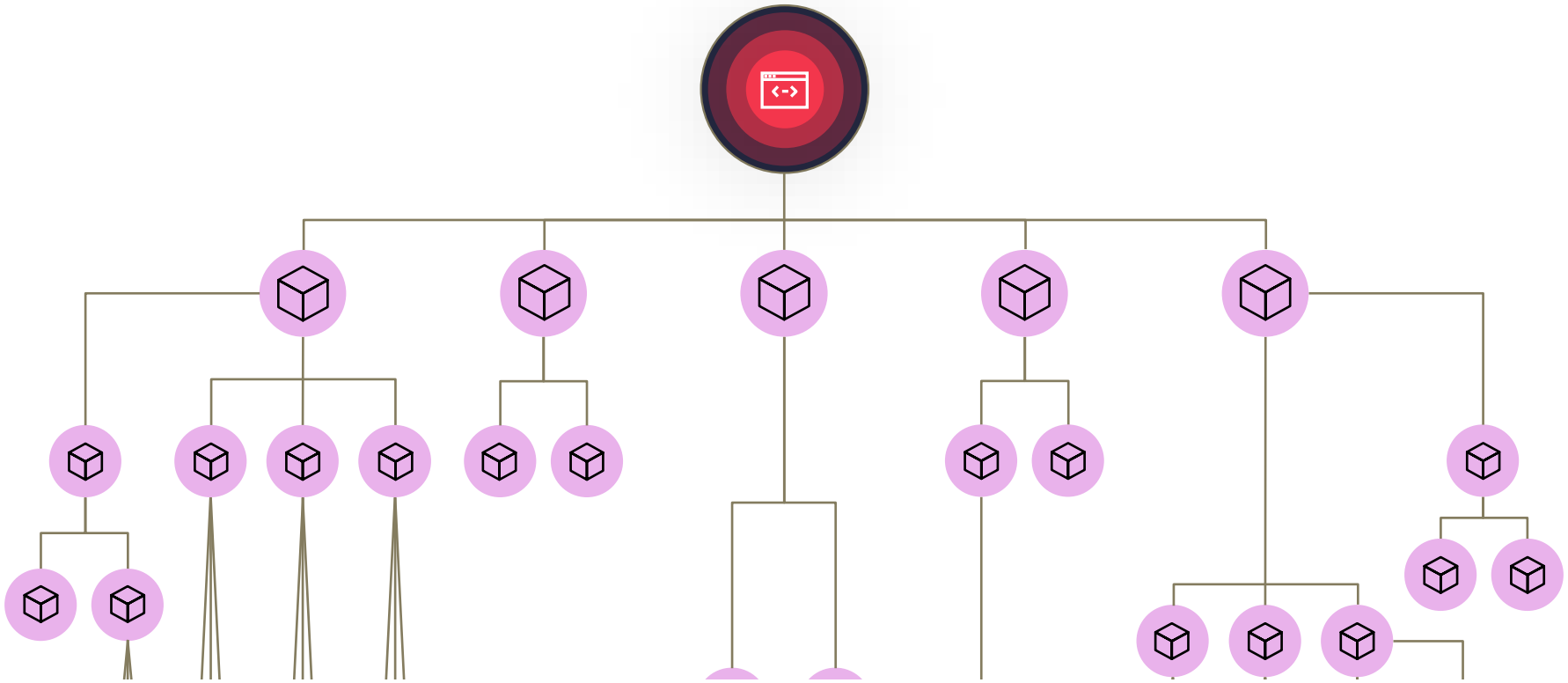


Azure SQL Database

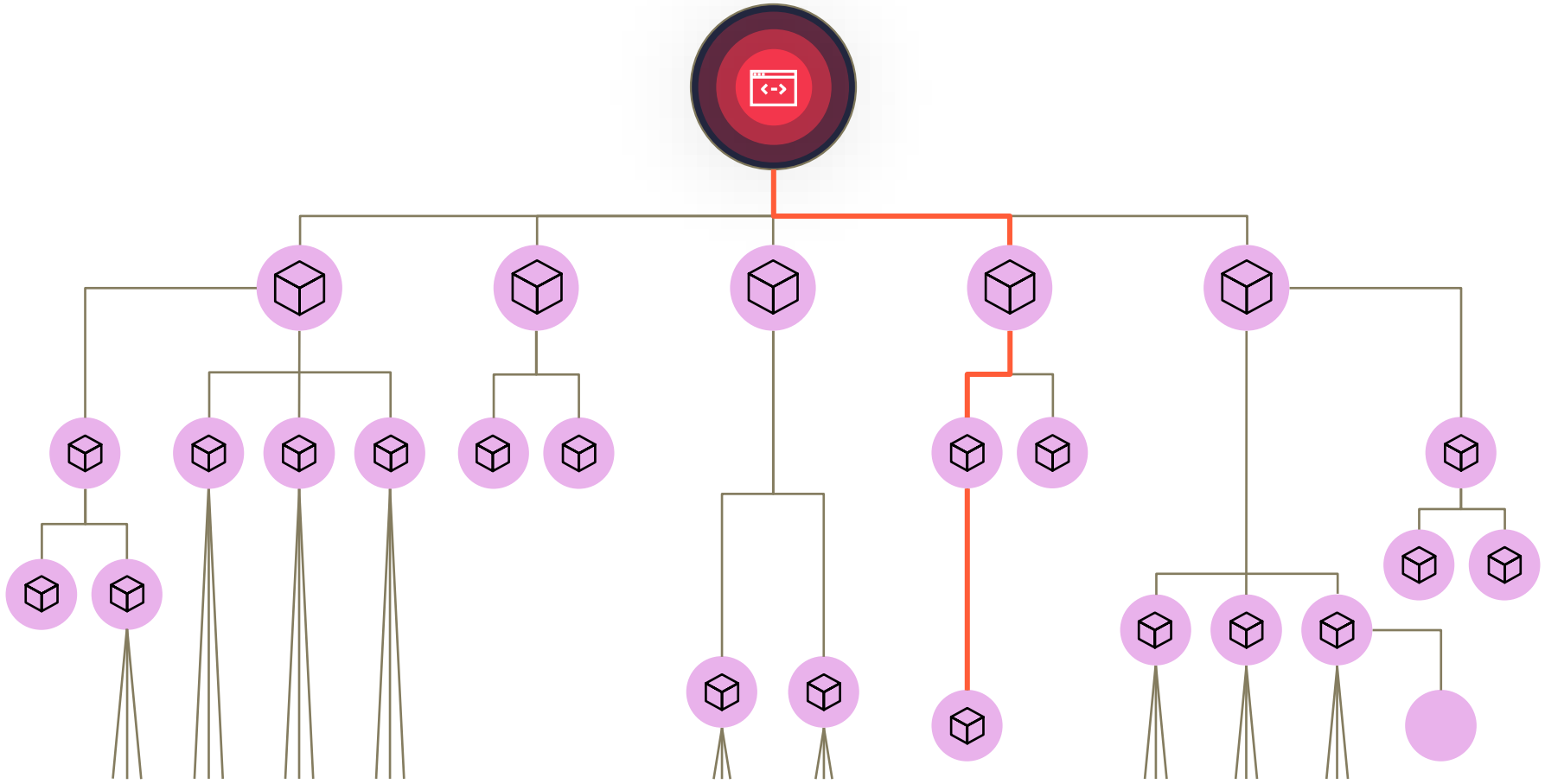
Azure Cosmos DB

Security and Vulnerability Management

Security and vulnerability management



Security and vulnerability management



Where security fits in the development lifecycle

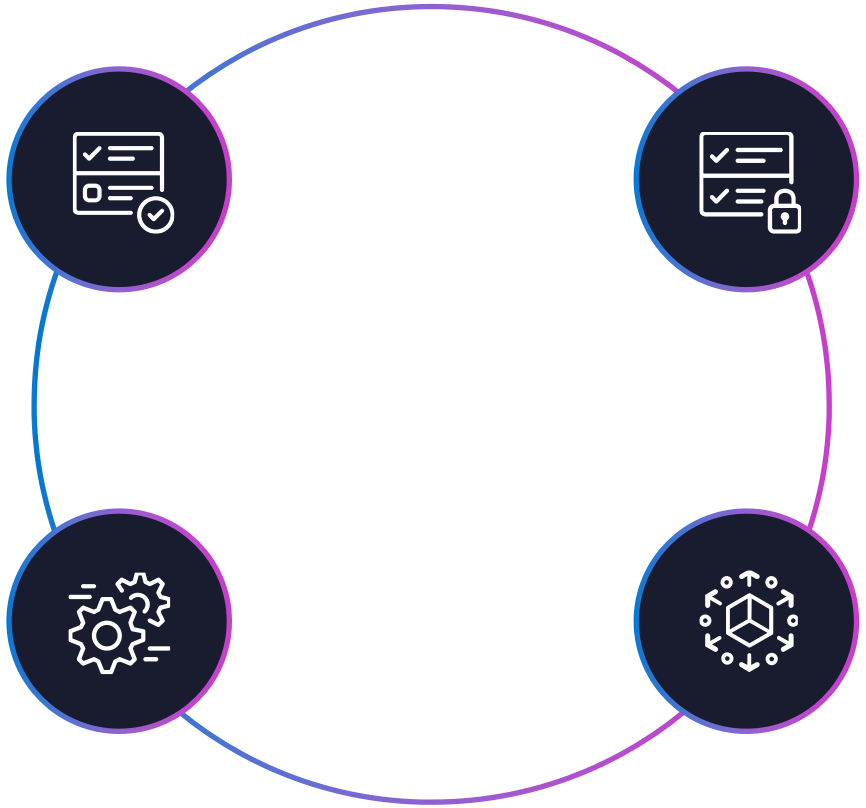
EMBEDED SECURITY IN THE DEVELOPER WORKFLOW

PRE-COMMIT

- Threat modeling
- IDE security plug-in
- Pre-commit hooks
- Secure coding standards
- Peer review

OPERATE & MONITOR

- Continuous monitoring
- Threat intelligence
- Blameless postmortems



COMMIT (CI)

- Static code analysis
- Security unit tests
- Dependency management
- Credential scanning

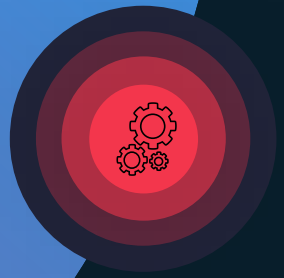
DEPLOY (CD)

- Infra as code (IaC)
- Security scanning
- Cloud configuration
- Security acceptance tests

What Tailwind Traders Needs

Dependency Insights

- Real-time inventory
- License compliance
- Vulnerability alerting



Vulnerability Management

- Code scanning
- Secret scanning
- Largest vulnerability database
- Automated security updates

CodeQL

- World's most advanced code analysis
- Vulnerability hunting tool
- Community of top security experts

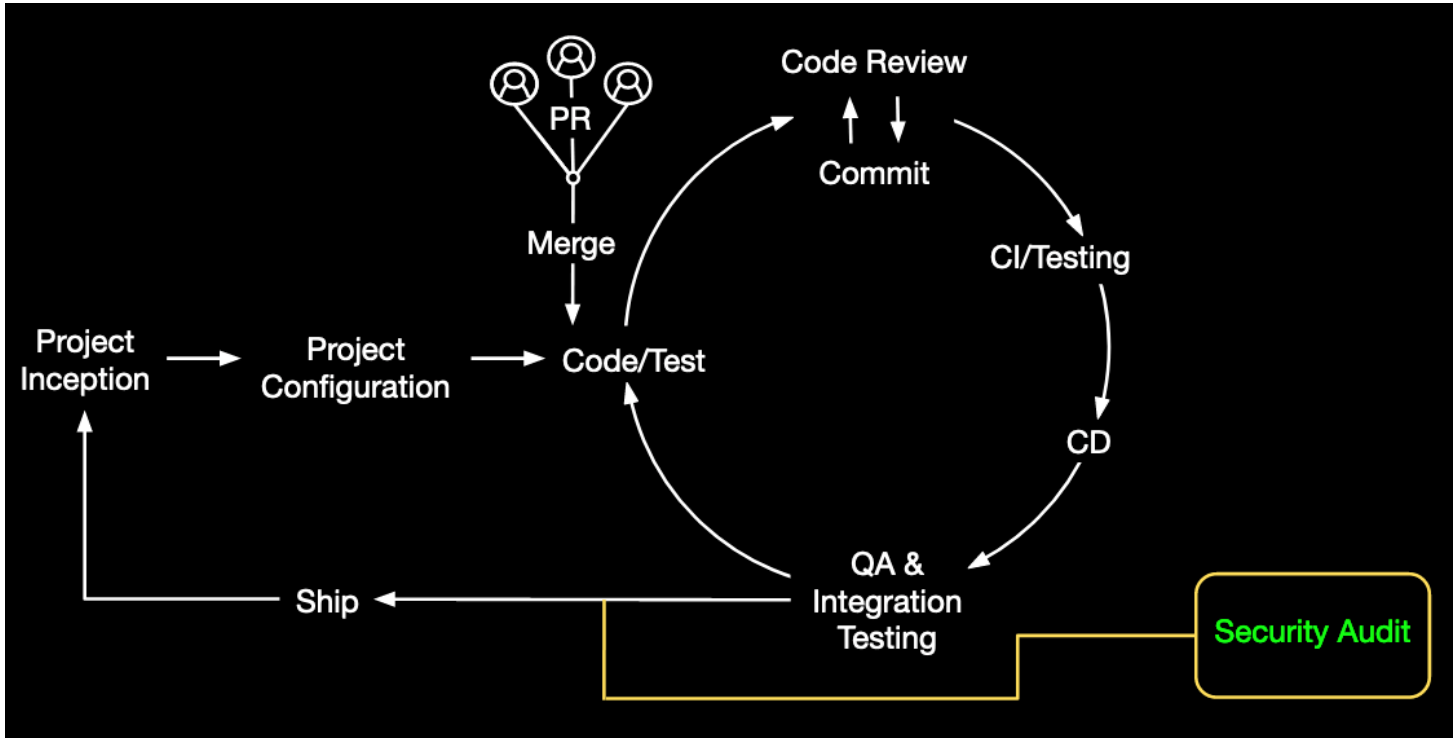
GitHub Advanced Security

Feature	Public repository	Private repository without Advanced Security	Private repository with Advanced Security
Code scanning	Yes	No	Yes
Secret scanning	Yes (limited functionality only)	No	Yes
Dependency review	Yes	No	Yes
Security Overview	No	No	Yes
Push Protection	No	No	Yes

- **Code scanning:** Automatically detect common vulnerabilities and coding errors.
- **Secret scanning:** Receive alerts when secrets or keys are checked in, exclude files from scanning, and define up to 100 custom patterns.
- **Dependency review:** Show the full impact of changes to dependencies and see details of any vulnerable versions before you merge a pull request.
- **Security Overview:** Review the security configuration and alerts for an organization and identify the repositories at greatest risk.
- **Push Protection:** Use secret scanning to prevent supported secrets from being pushed into your organization or repository.

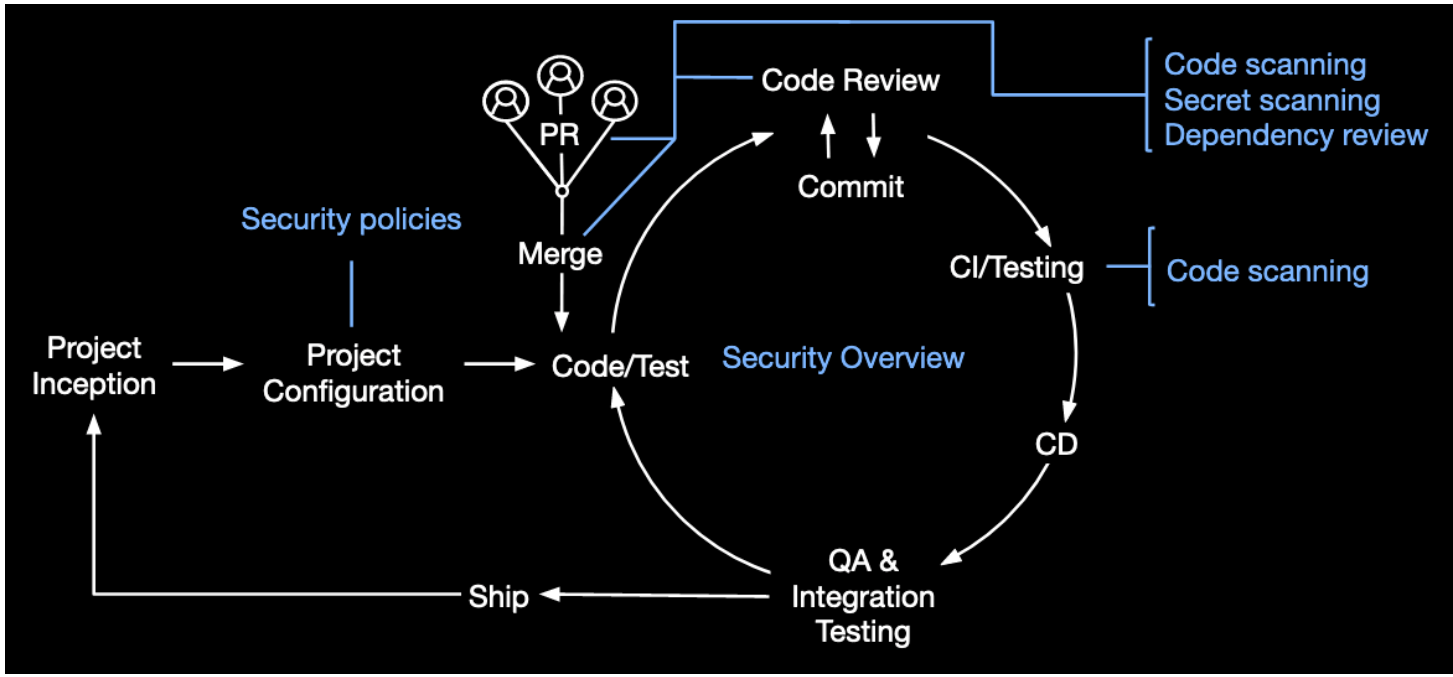
GitHub Advanced Security in the software development lifecycle

Traditional “security as a gate” approach.



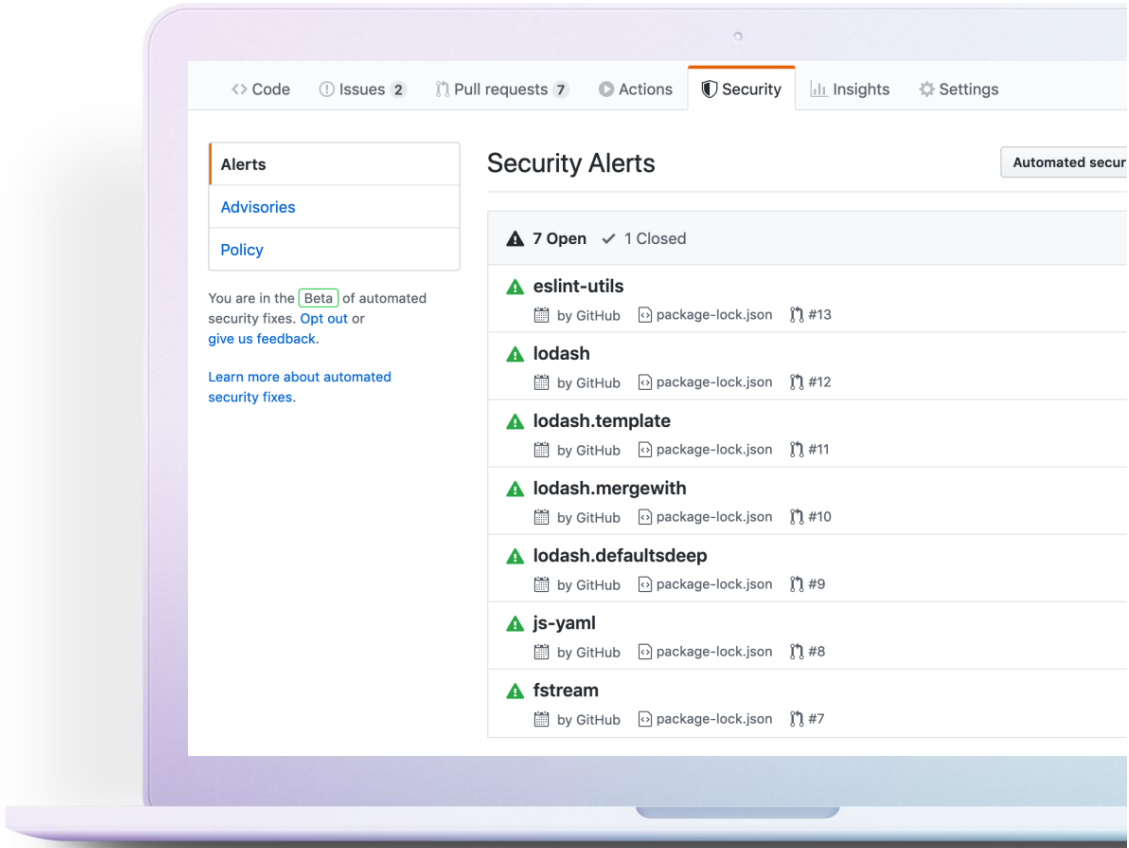
GitHub Advanced Security in the software development lifecycle

Software development lifecycle with GitHub Advanced Security.



Vulnerability Management

Over 62 million security alerts sent across GitHub.



Code Scanning

Find and fix vulnerabilities fast

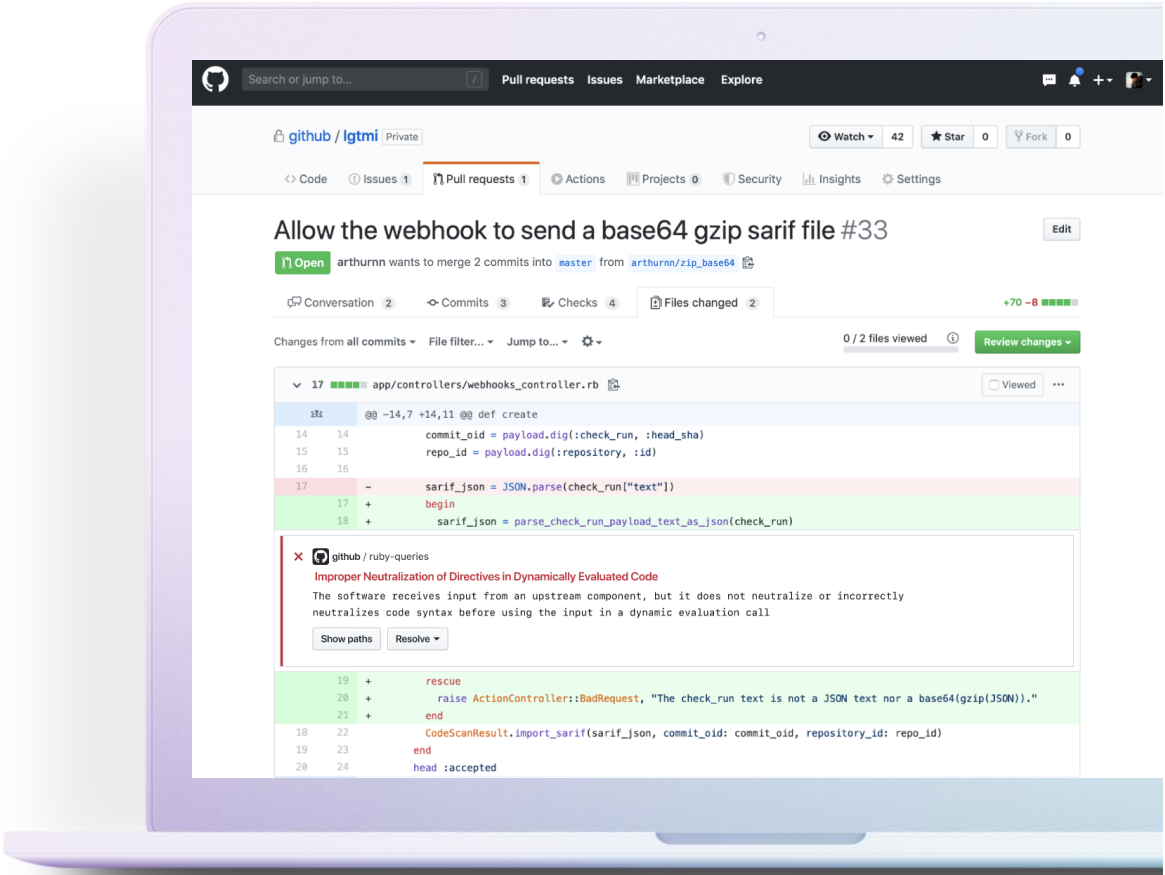
Find and fix vulnerabilities before they are merged into the code base with automated CodeQL scans

Community of top security experts

Community-driven query set powers every project with a world-class security team

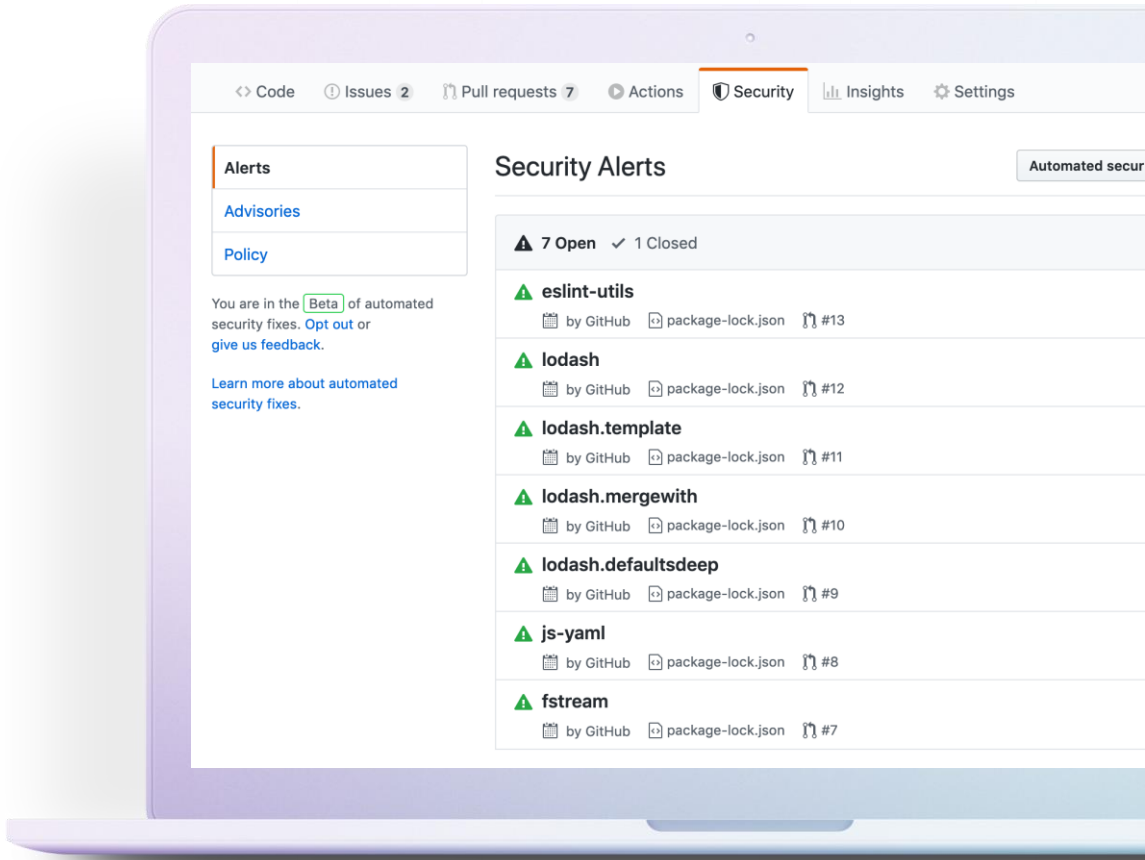
Integrated with developer workflow

Integrate security results directly into the developer workflow for a frictionless experience and faster development



Secret Scanning

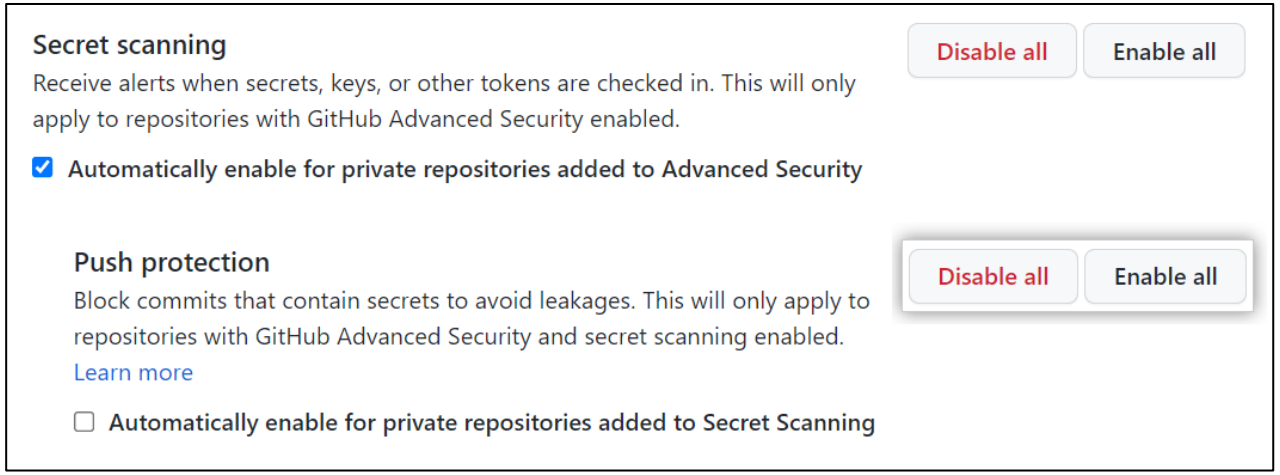
Scan for leaked secrets in public and private repos.



GitHub Secret Scanning Push Protection

- By enabling **push protection**, you can use secret scanning to prevent supported secrets from being pushed into your organization or repository.
- Secret scanning lists any secrets it detects so the author can review the secrets and remove them or, allow those secrets to be pushed.

- If a contributor **bypasses** a push protection block for a secret, GitHub:
  - Generates an alert.
  - Creates an alert in the "Security" tab of the repository.
  - Adds the bypass event to the audit log.
  - Sends an email alert with a link to the related secret and the reason why it was allowed.

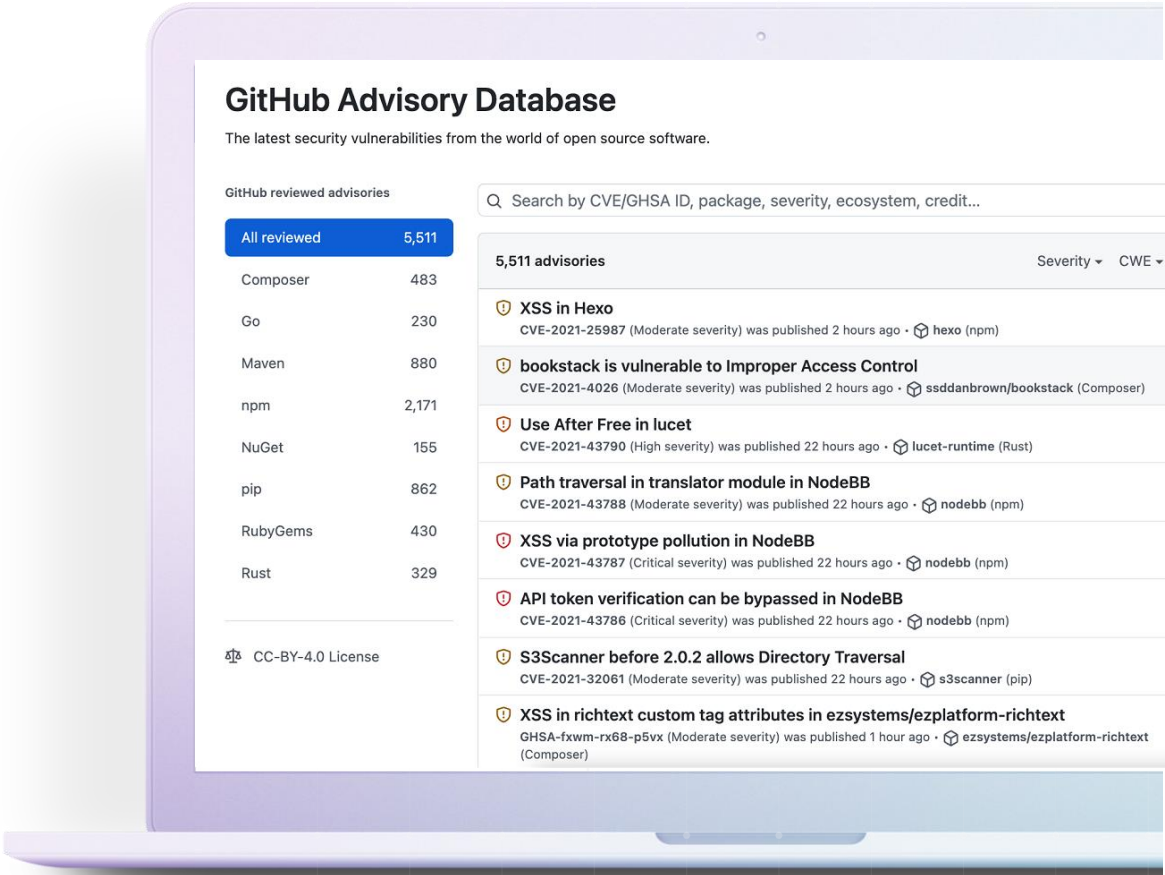


GitHub Advisory Database

The GitHub Advisory Database provides information on the state of your dependencies.

Search the GitHub Advisory Database for vulnerabilities in third-party solutions.

List security vulnerabilities mapped to packages tracked by dependency graphs.



Dependabot

Check for updates

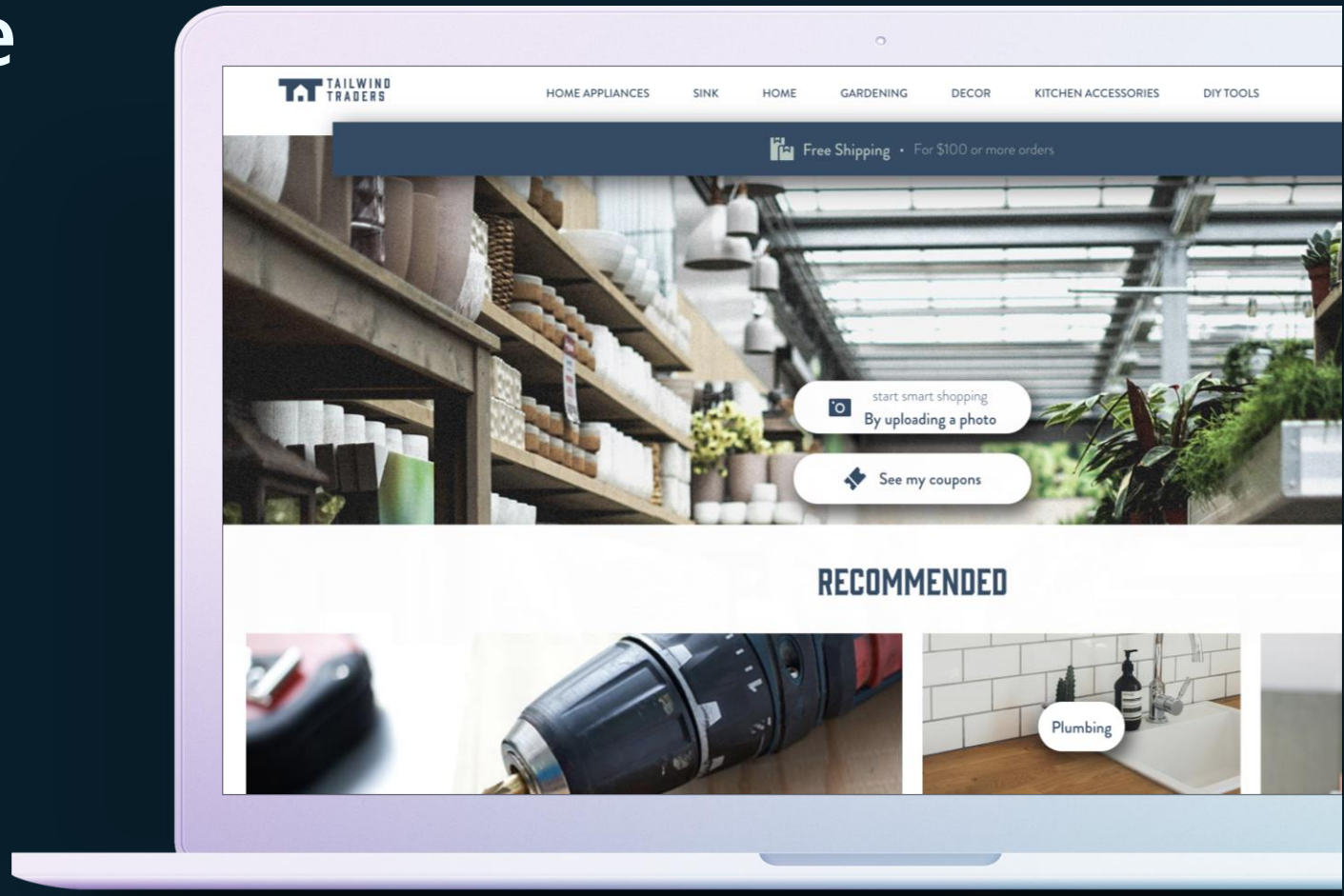
Alerts

Automated pull requests

Review and triage

Tailwind Traders Website

- Vulnerable container images
- Missing Settings
- Passwords
- Database Connection strings
- Secrets



GitHub security

Code Scanning

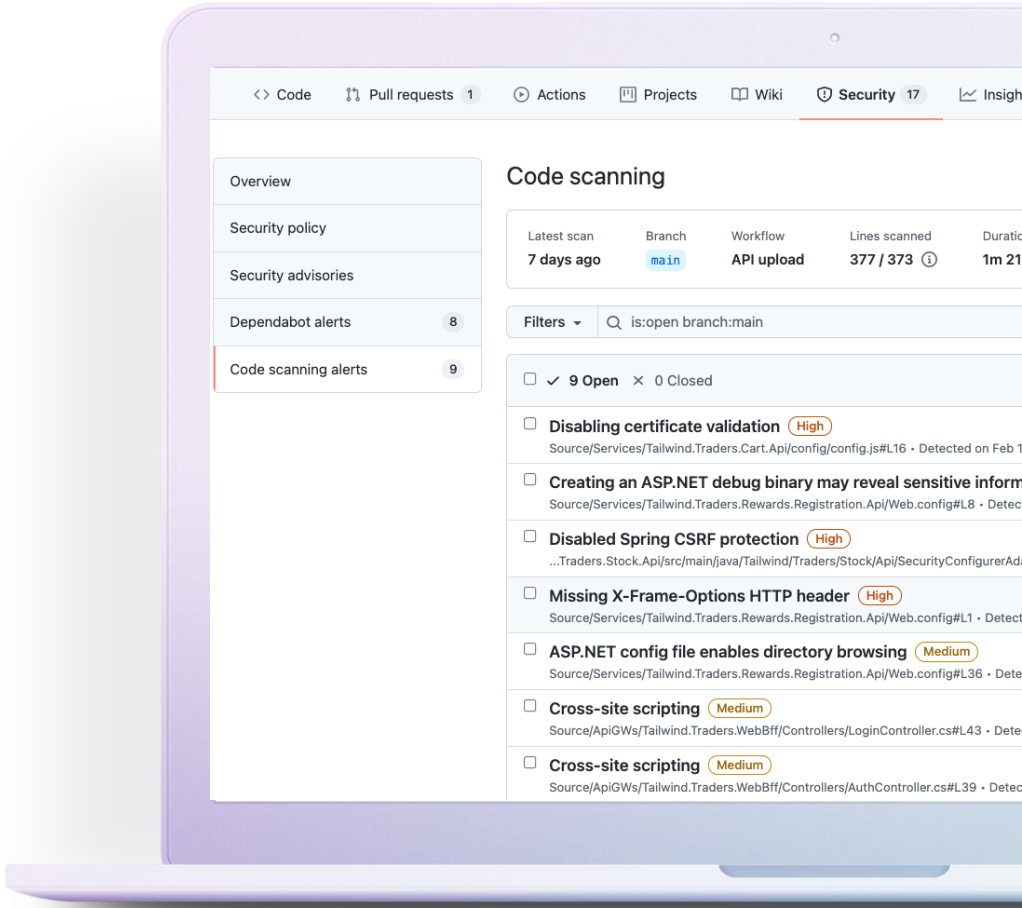
Find and fix vulnerabilities before they are merged into the code base with automated CodeQL scans

Scan containers

Scan for common vulnerabilities in Docker images before pushing them to a container registry or deploying them to a containerized web app or Kubernetes cluster

Manage secrets using Azure Key Vault

Dynamically pull secrets from an Azure Key Vault instance for consumption in GitHub Action workflows

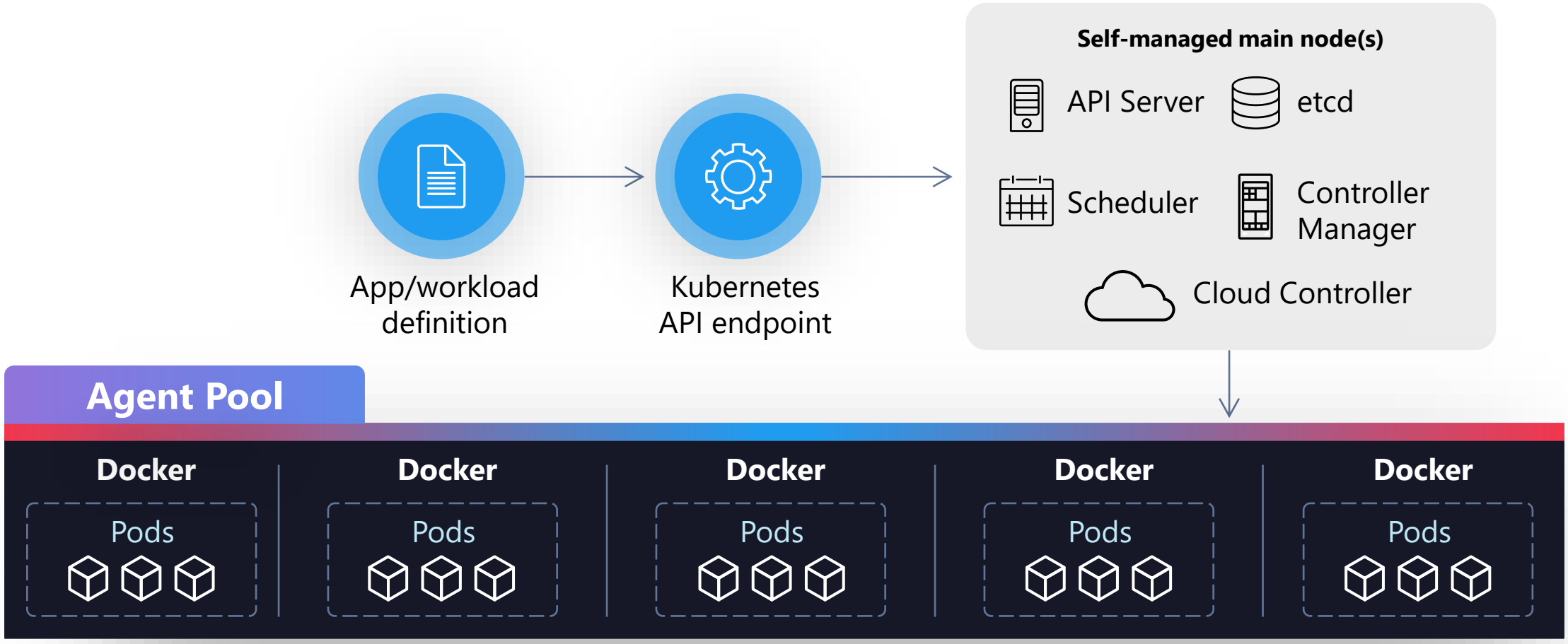


Demo

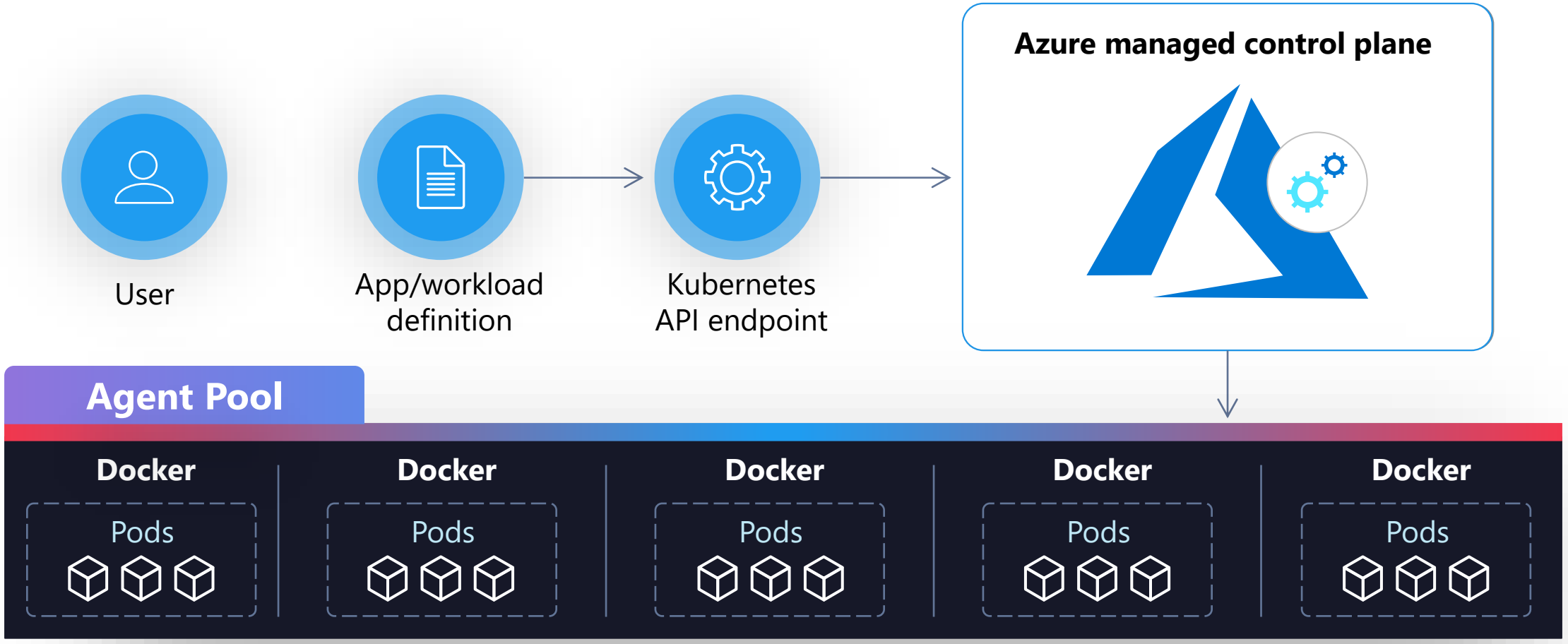
Azure Security and GitHub Advanced Security

Container Security

Kubernetes architecture



Kubernetes Architecture



Refresher on container layers

Container Layer

91e49dfb1179

d7b1189bf667

c220123c8472

d31af33eb855

a7183fb762a8

f61792ba8979

• • •

From: Alpine:3.8

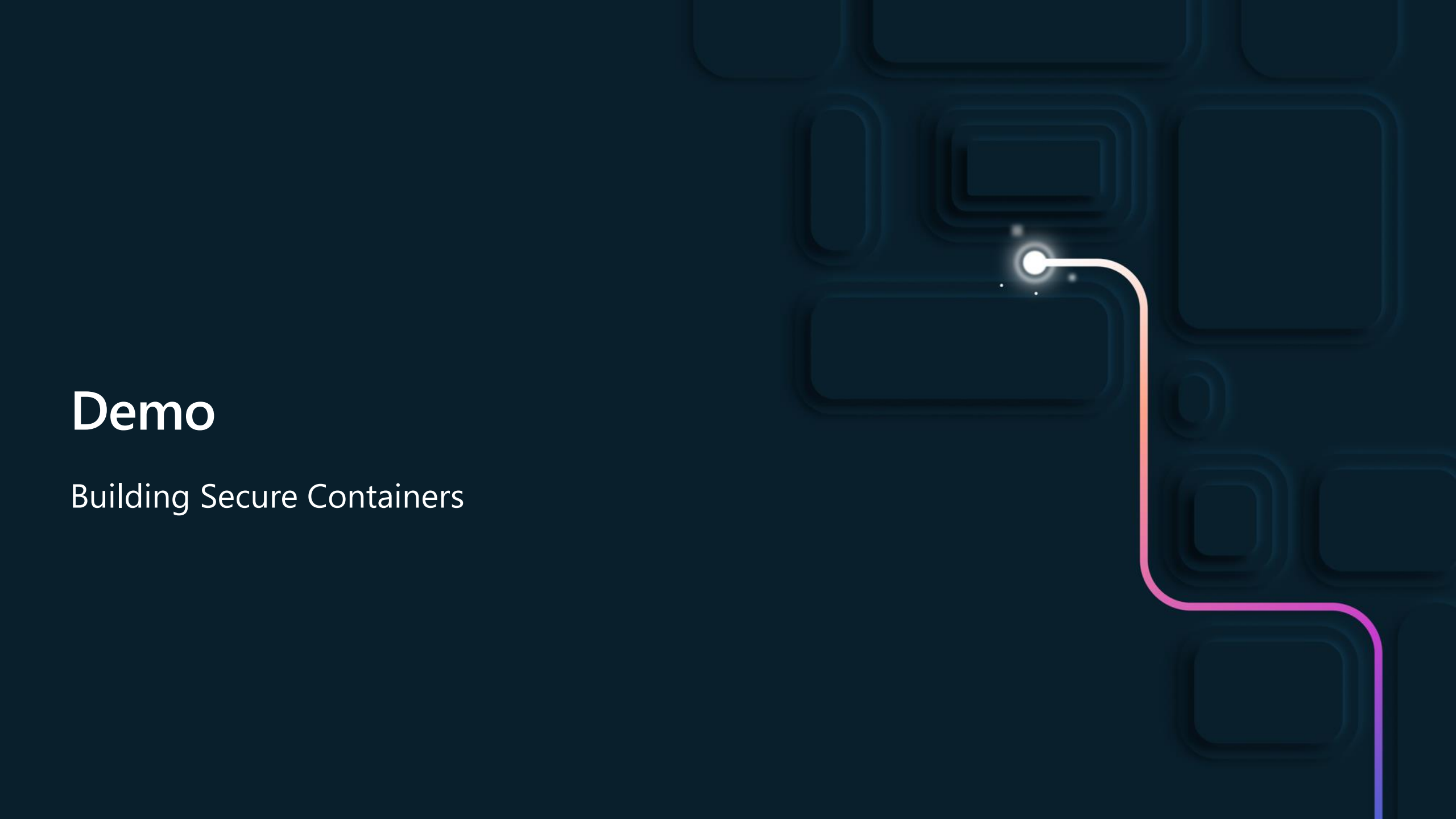
Read/Write

Image Layers

Read Only

Demo

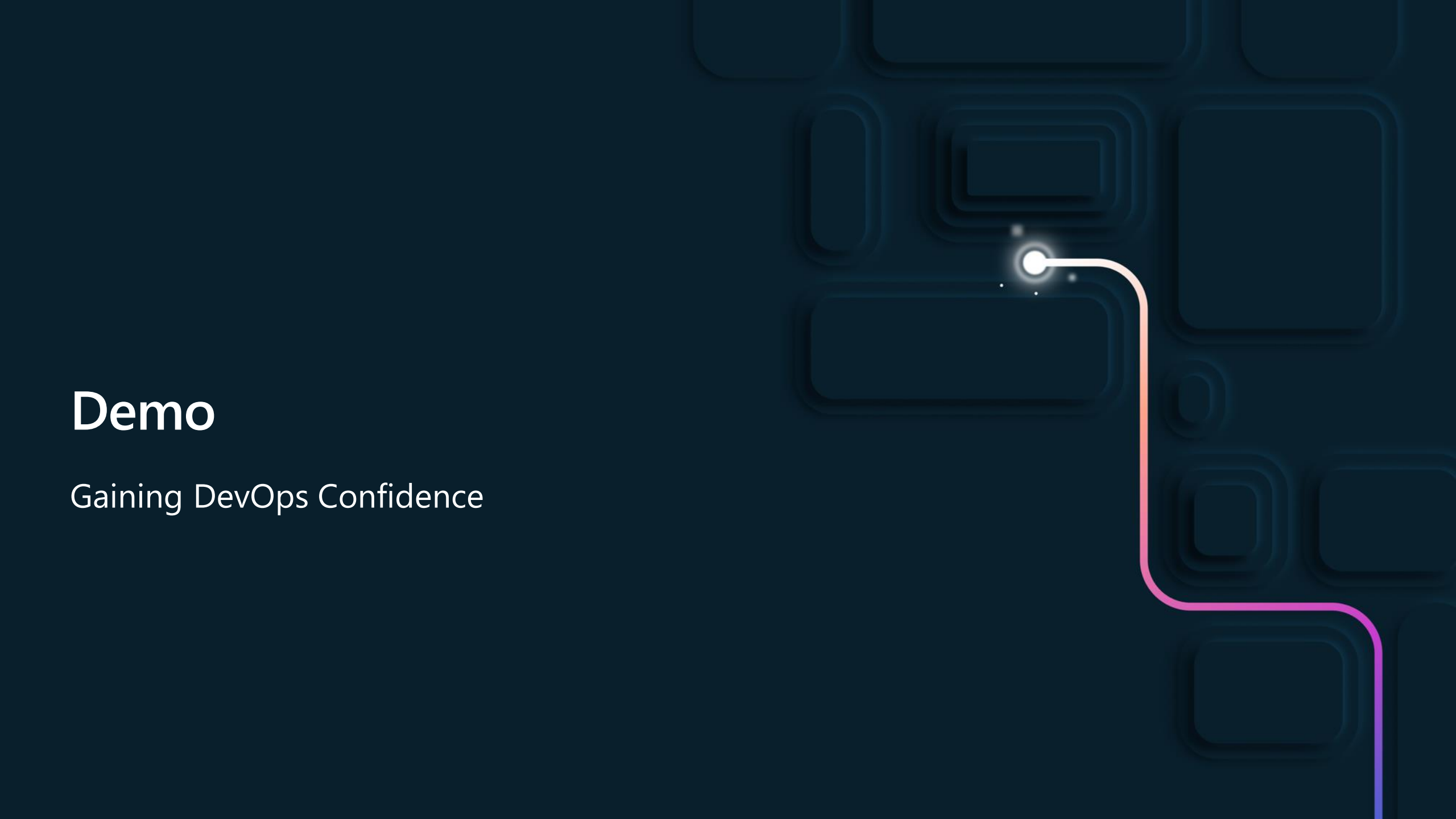
Building Secure Containers



Quality

Demo

Gaining DevOps Confidence



Bridge to Kubernetes

Simplifies microservice development

Eliminates the need to manually source, configure, and compile external dependencies

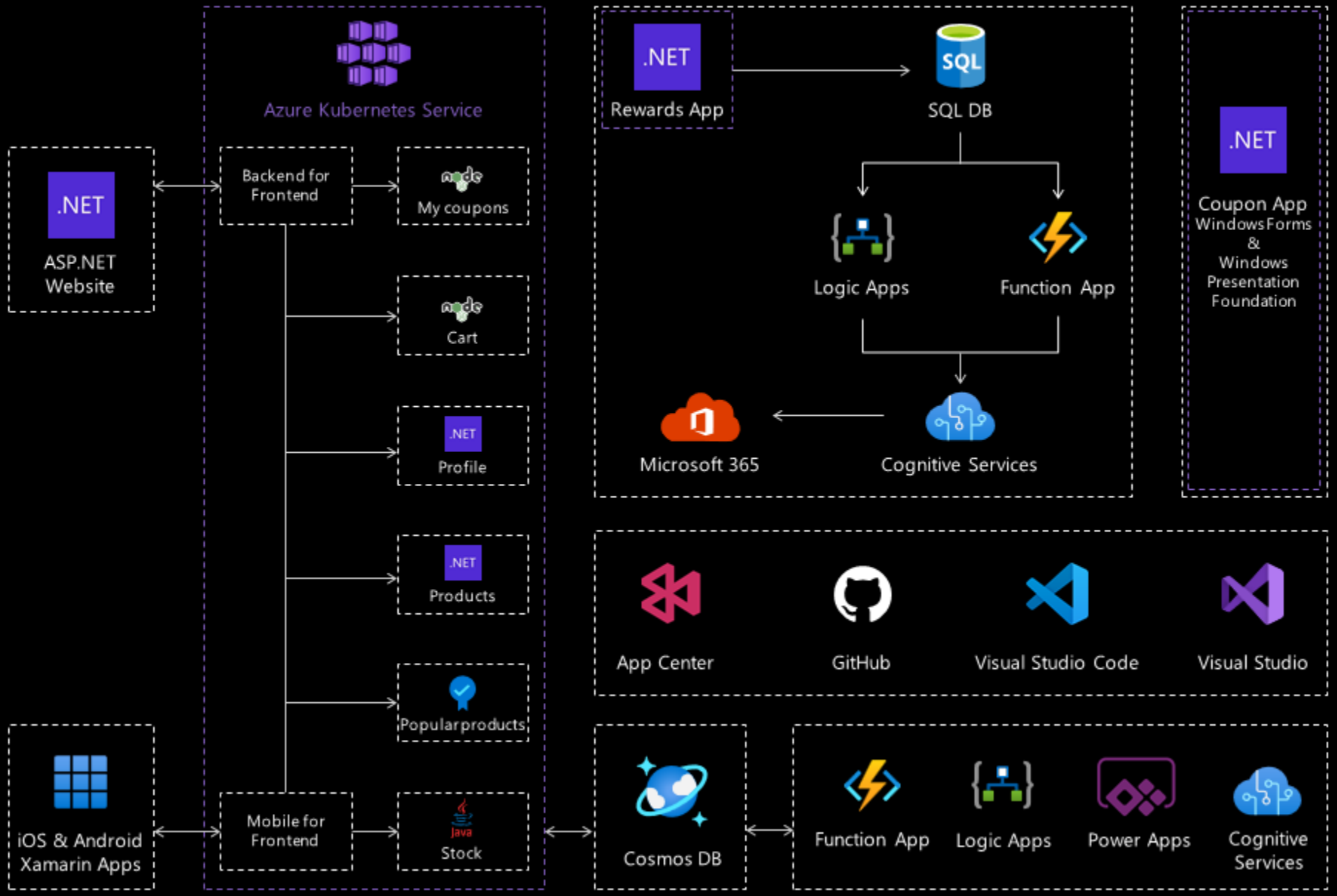
Streamlines application development

Sidestep operational complexities of building and deploying code into the cluster to test and debug

Work in isolation in shared development environment

Work in a private “sandbox” environment by routing specific traffic locally

TAILWIND TRADERS

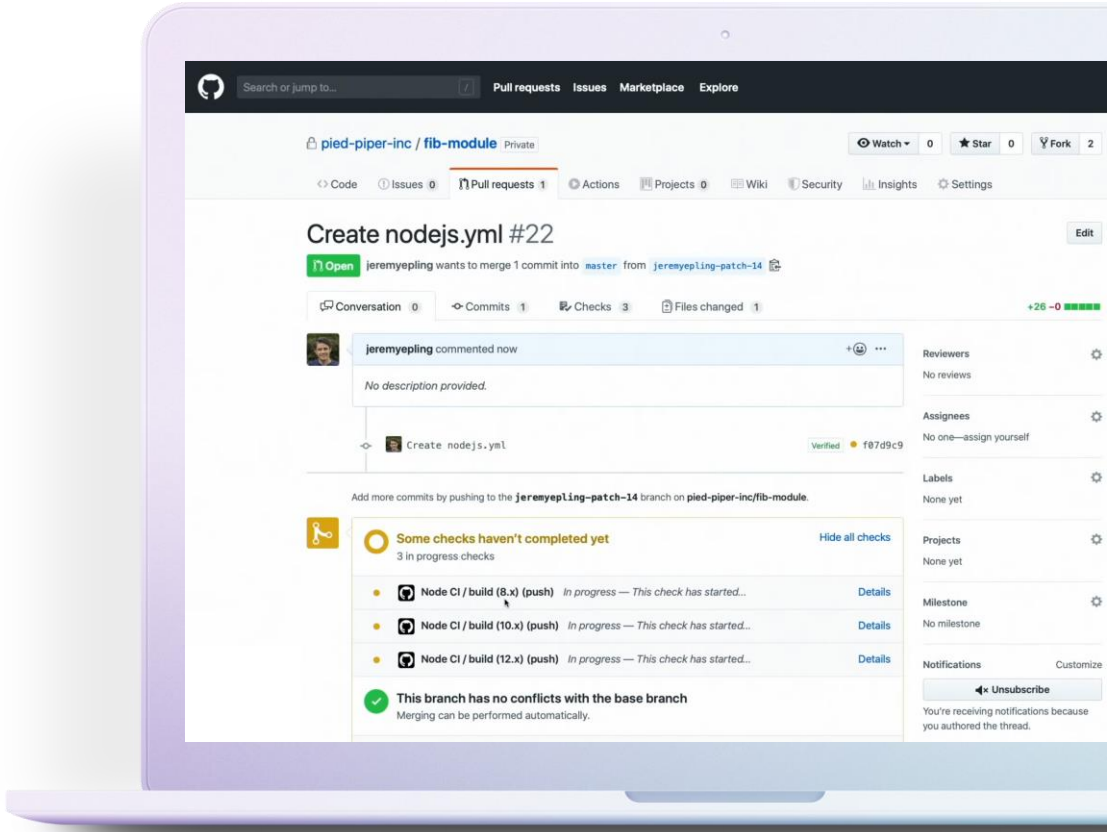


GitHub Actions

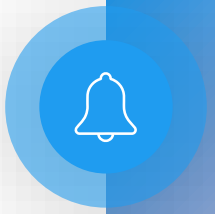
Automate

Build, test and deploy with confidence

Customizable

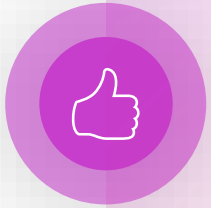


What did we learn?

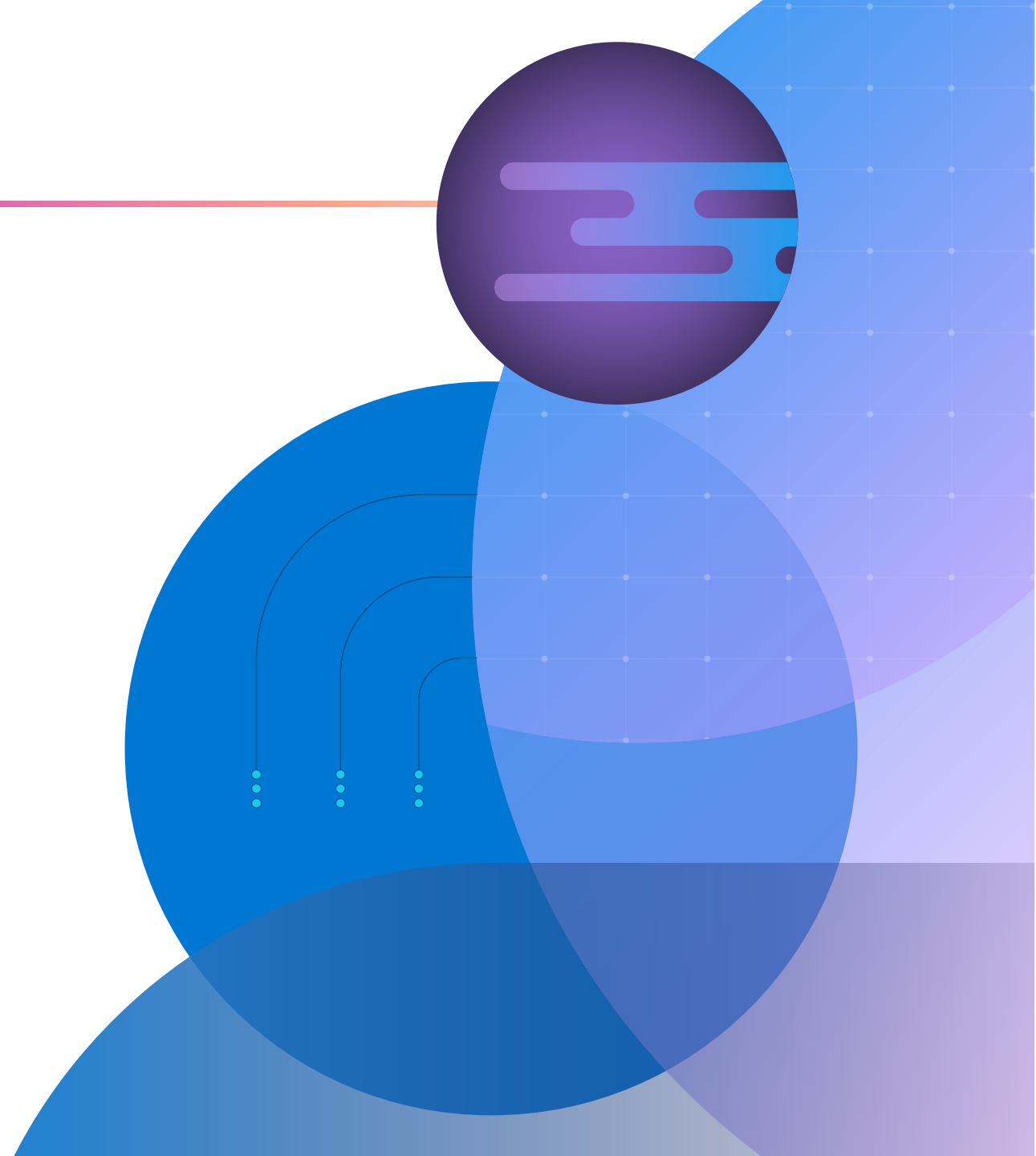


Code Scanning and Dependency Alerts

Build More Secure Containers



Gain DevOps Confidence



Microsoft

Invent with purpose.

Microsoft Azure

Microsoft

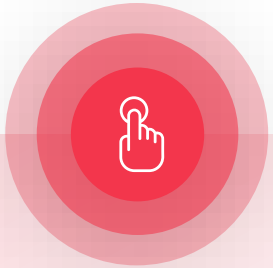
Delivering Changes to the Cloud

DevOps Learning Path

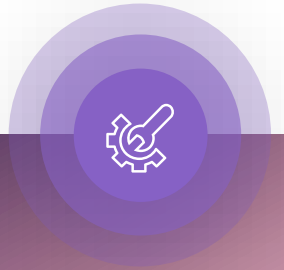
- Getting Started with DevOps
- Managing the Flow of Work
- Shift security “left” in your CI/CD process
- **Delivering changes to cloud**
- Performance Monitoring and maintenance

Goals for this Session

Continuous Integration
Continuous Delivery /
Deployment

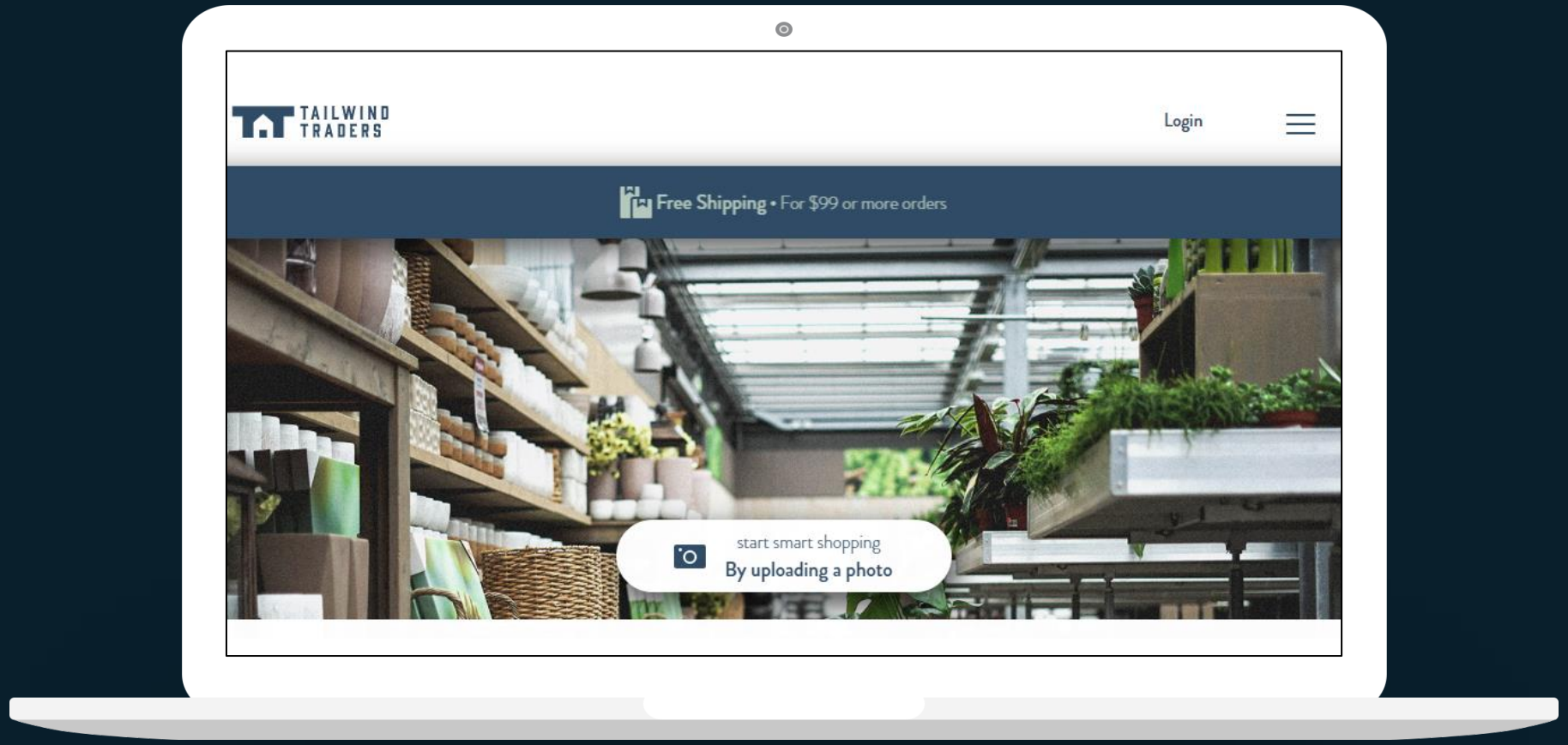


Trunk Based
Development



Protect Production /
Secrets

Tailwind Traders

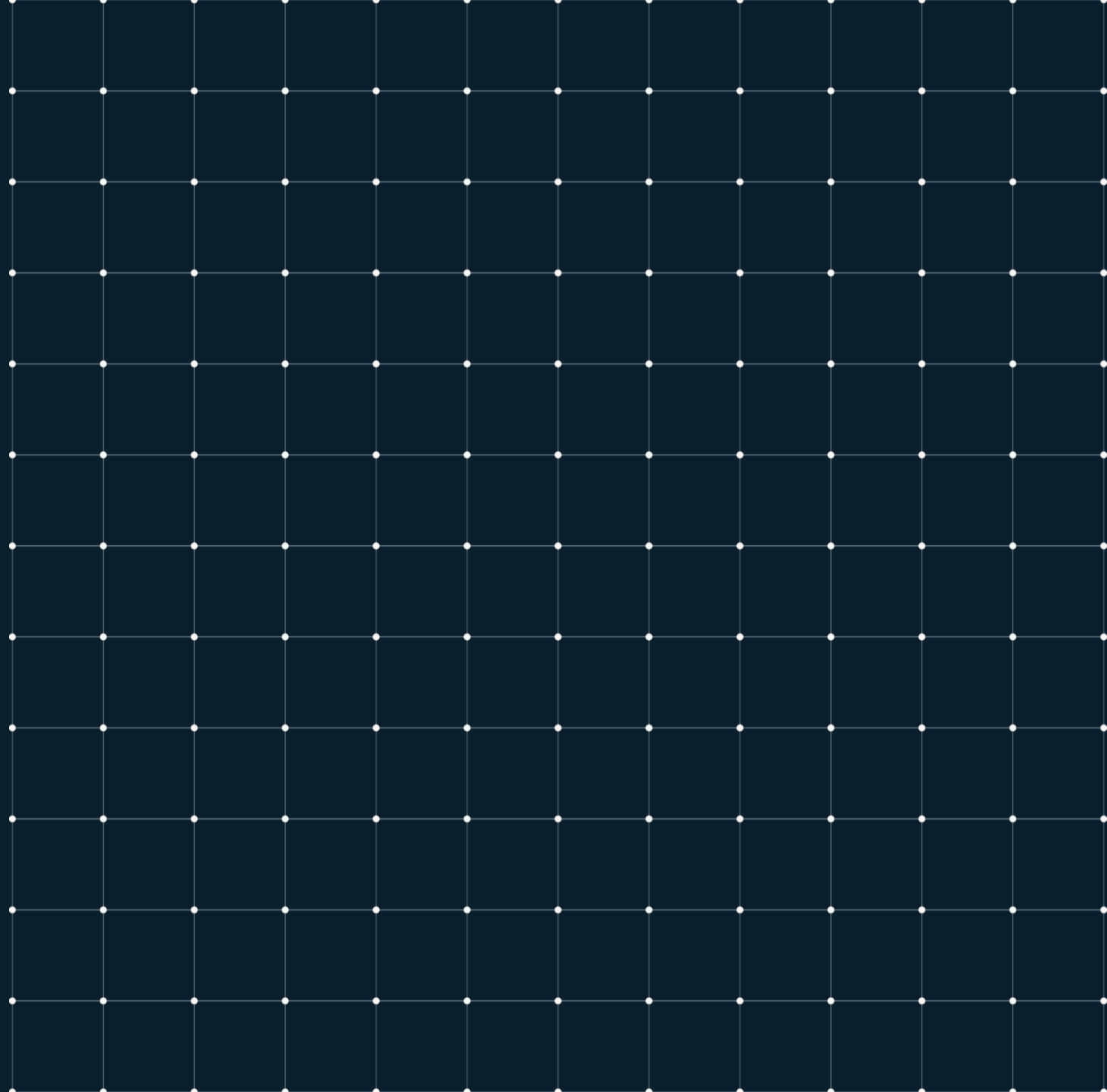


CI and CD

Continuous Integration

Continuous Delivery

Continuous Deployment



Continuous Integration

Your changes work with everyone else's changes

Your code still builds

Your tests still run

Continuous Delivery

You have a deployable piece of work

Including infrastructure and dependencies

Deploy from build to a testing, staging, and/or production environment

Continuous Deployment

Deploy that piece of work

Doesn’t have to be to Production

Trustworthy and reproducible

CI and CD

Continuous Integration –
Develop phase. Build, test, and validate code.

Continuous Delivery –

Automates delivery. New build artifact is available, artifact is deployed.

Continuous Deployment –

From when you commit and check in code to production, everything is automated.

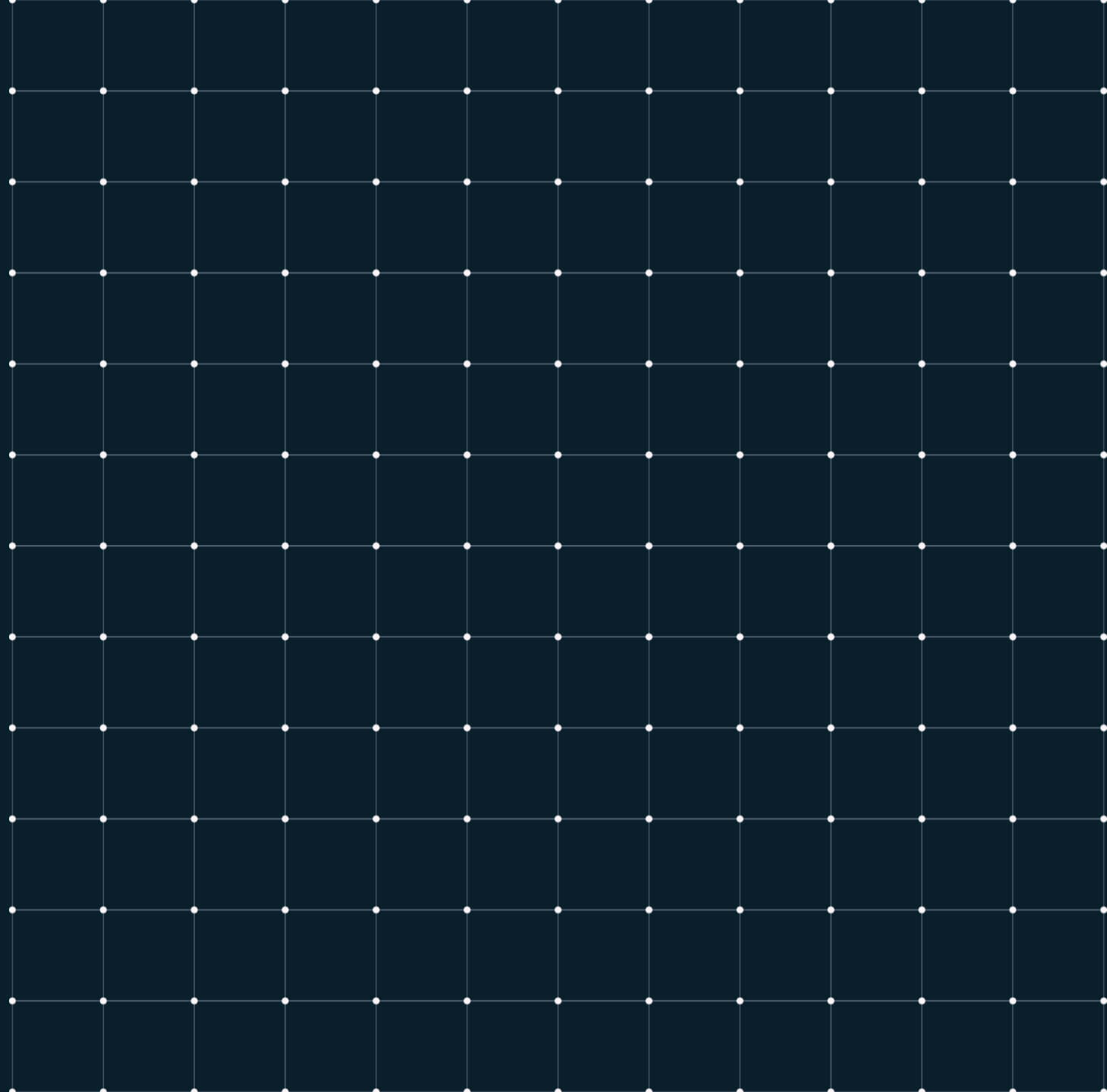
Protip

Always have Continuous Deployment to *somewhere*.

Don’t assume this version will deploy as cleanly as the last.

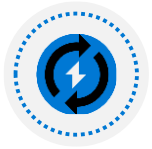
GitHub Actions

CI/CD with GitHub Actions

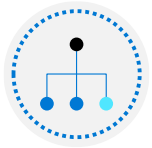


What are Actions?

Automations within the GitHub environment



Often used to build CI/CD implementations



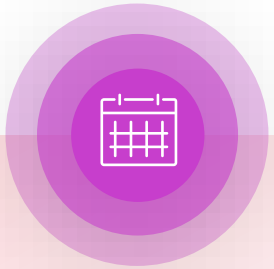
Based on YAML files living within GitHub repositories

Executed on GitHub or self-hosted runners

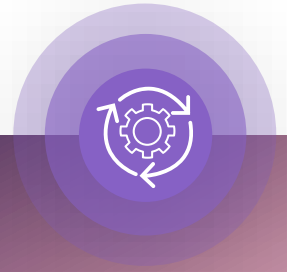
Large number of existing actions in the GitHub Marketplace

GitHub Actions

Automation for any software workflow



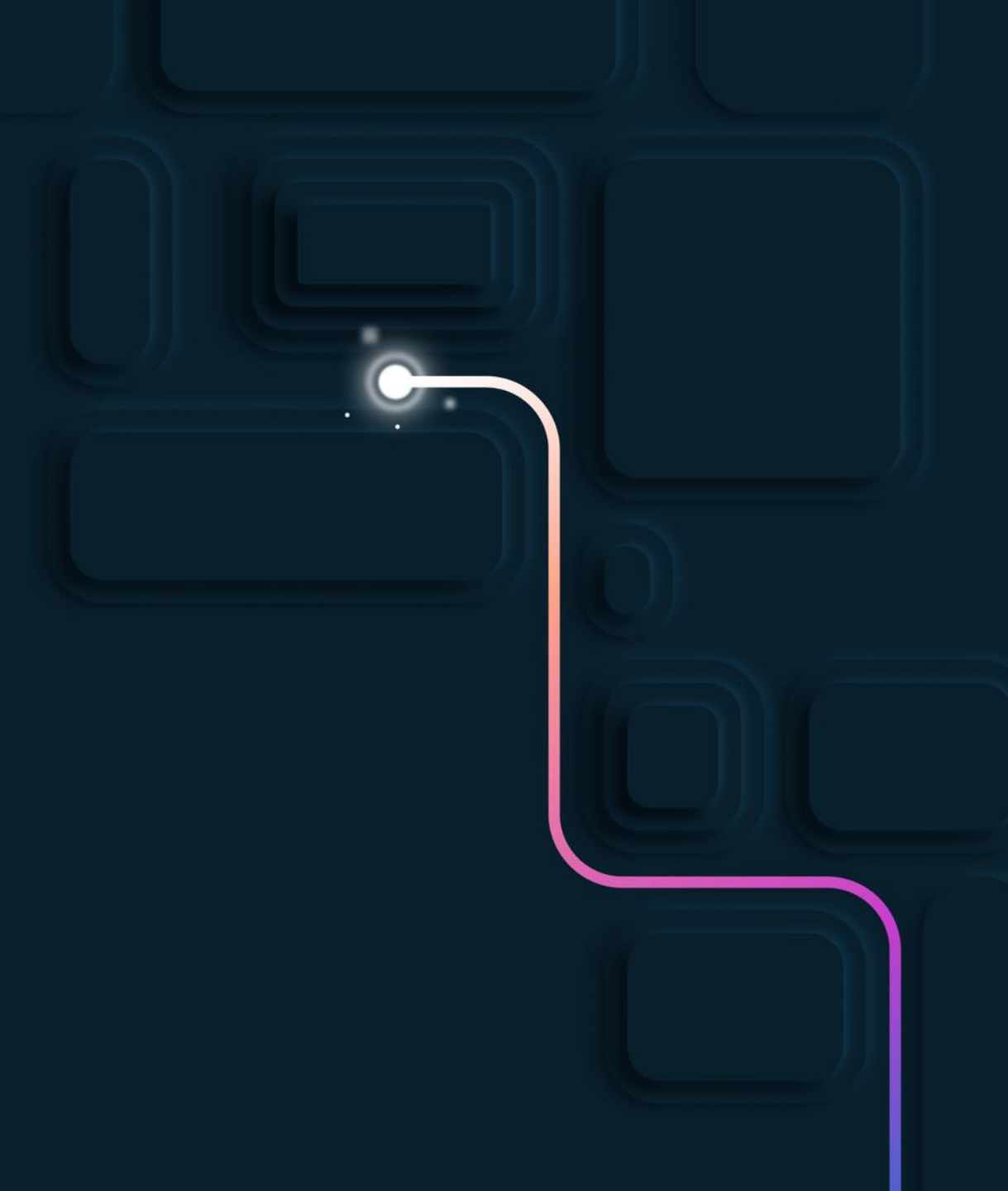
Dozens of events that can trigger workflows



Continuous Integration and Continuous Delivery

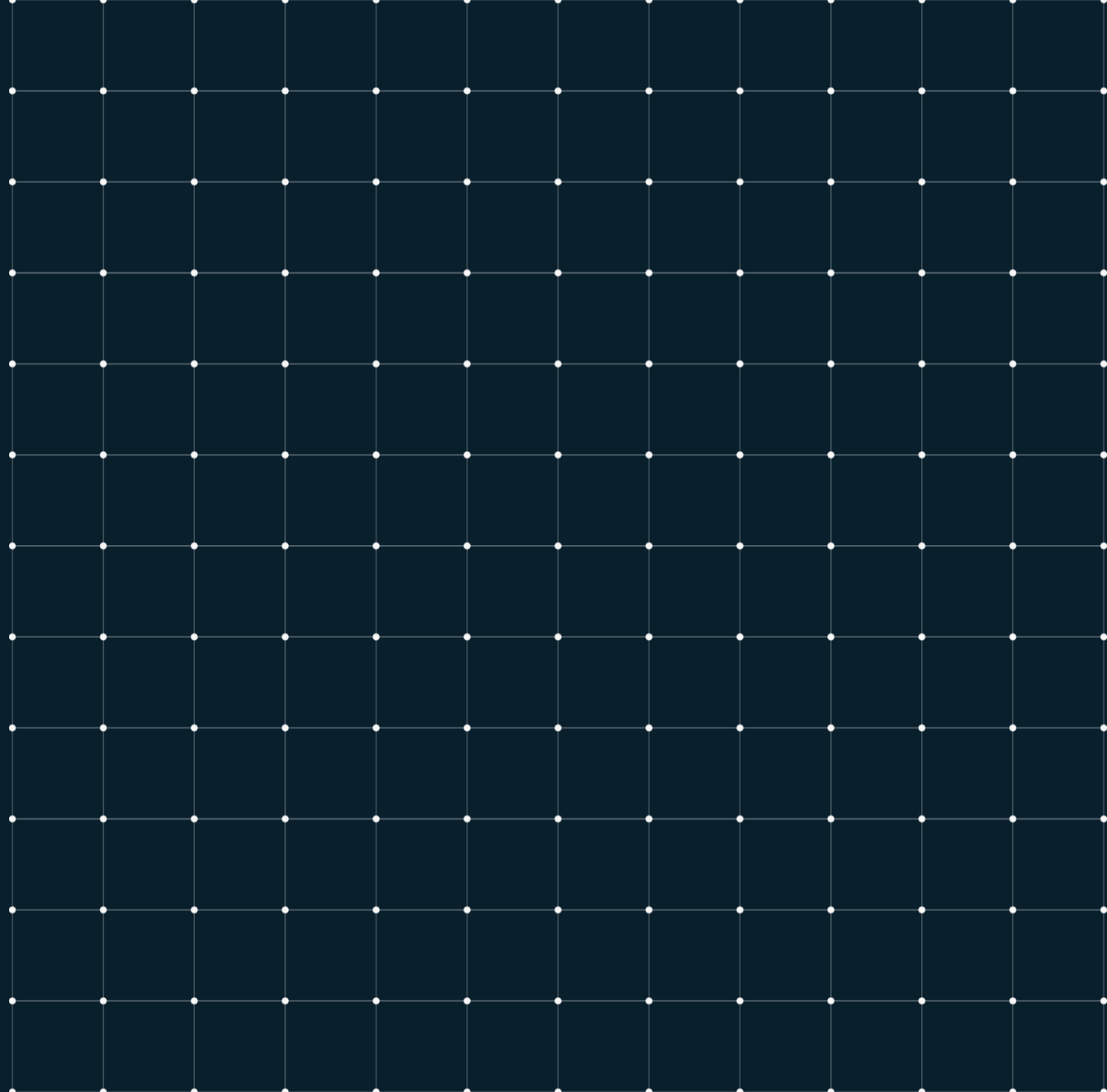
Demo

GitHub Actions – CI/CD

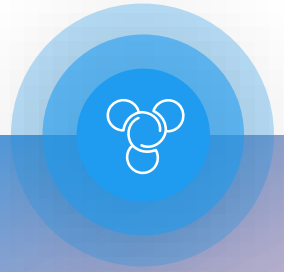


Protecting Production

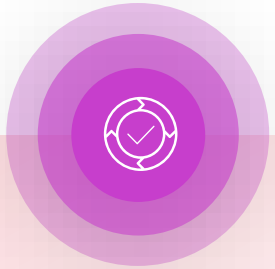
We might not want every change to go to Production



Pull Request Workflows



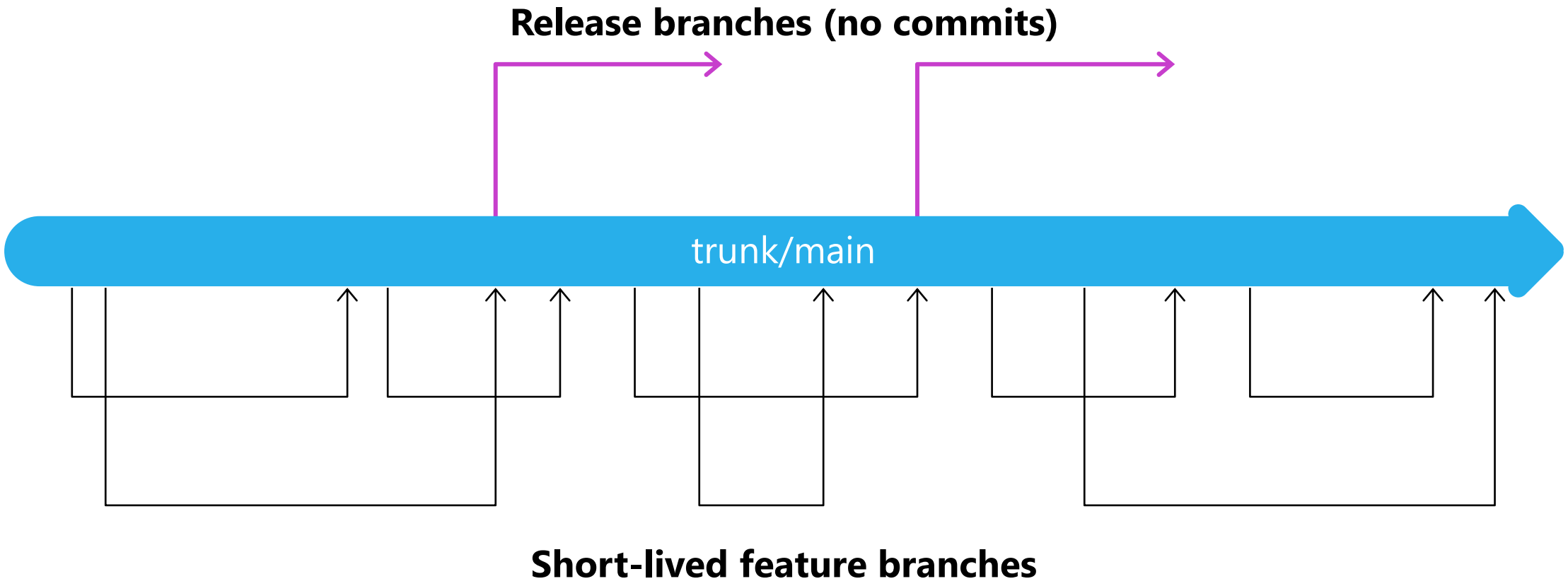
Workflow triggers on PR



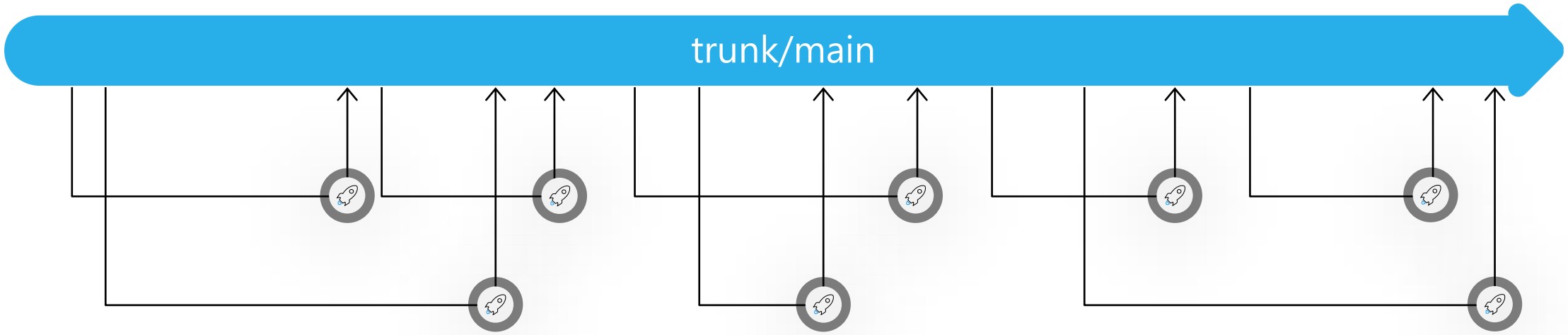
Build, test, deploy

Checks before merging to main branch

Trunk-based development

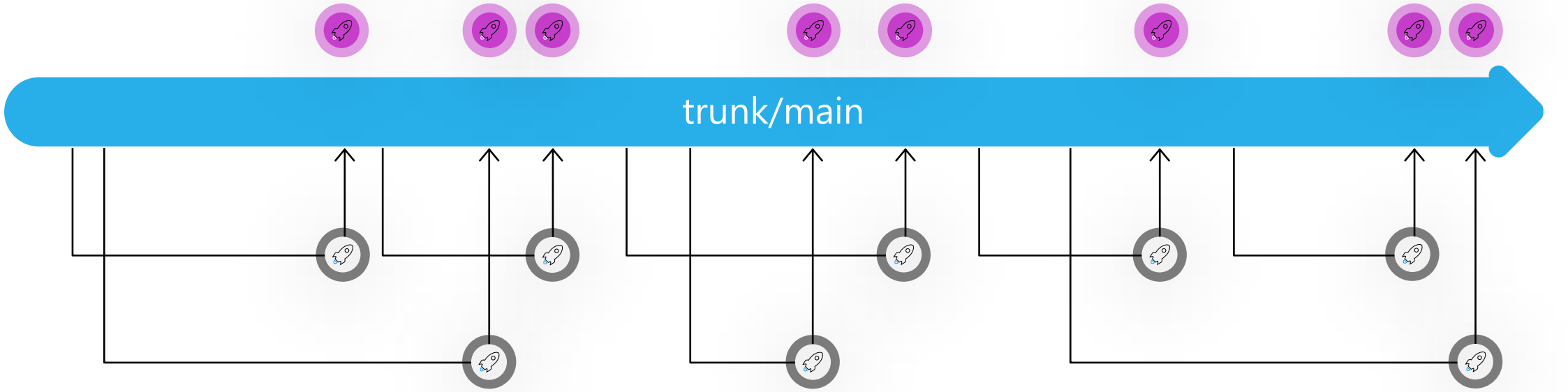


Trunk-based development – Topic Branches



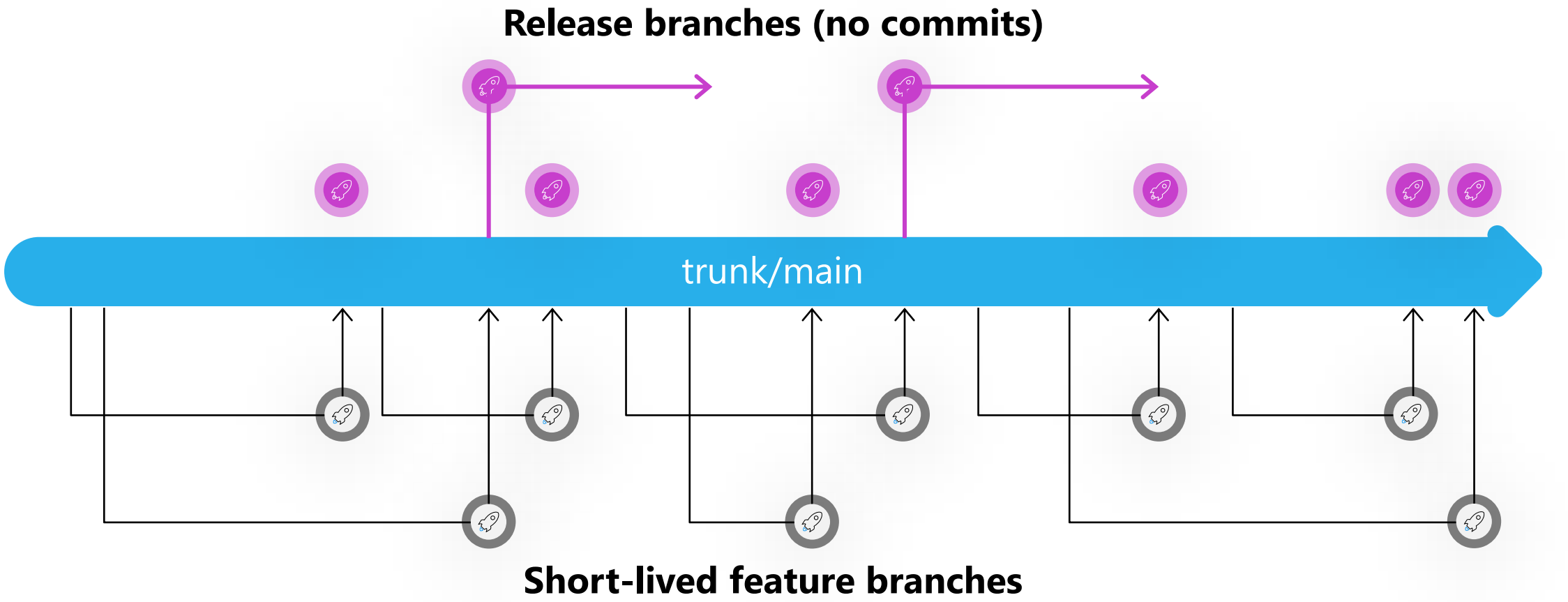
Short-lived feature branches

Trunk-based development – Merge, Build and Deploy



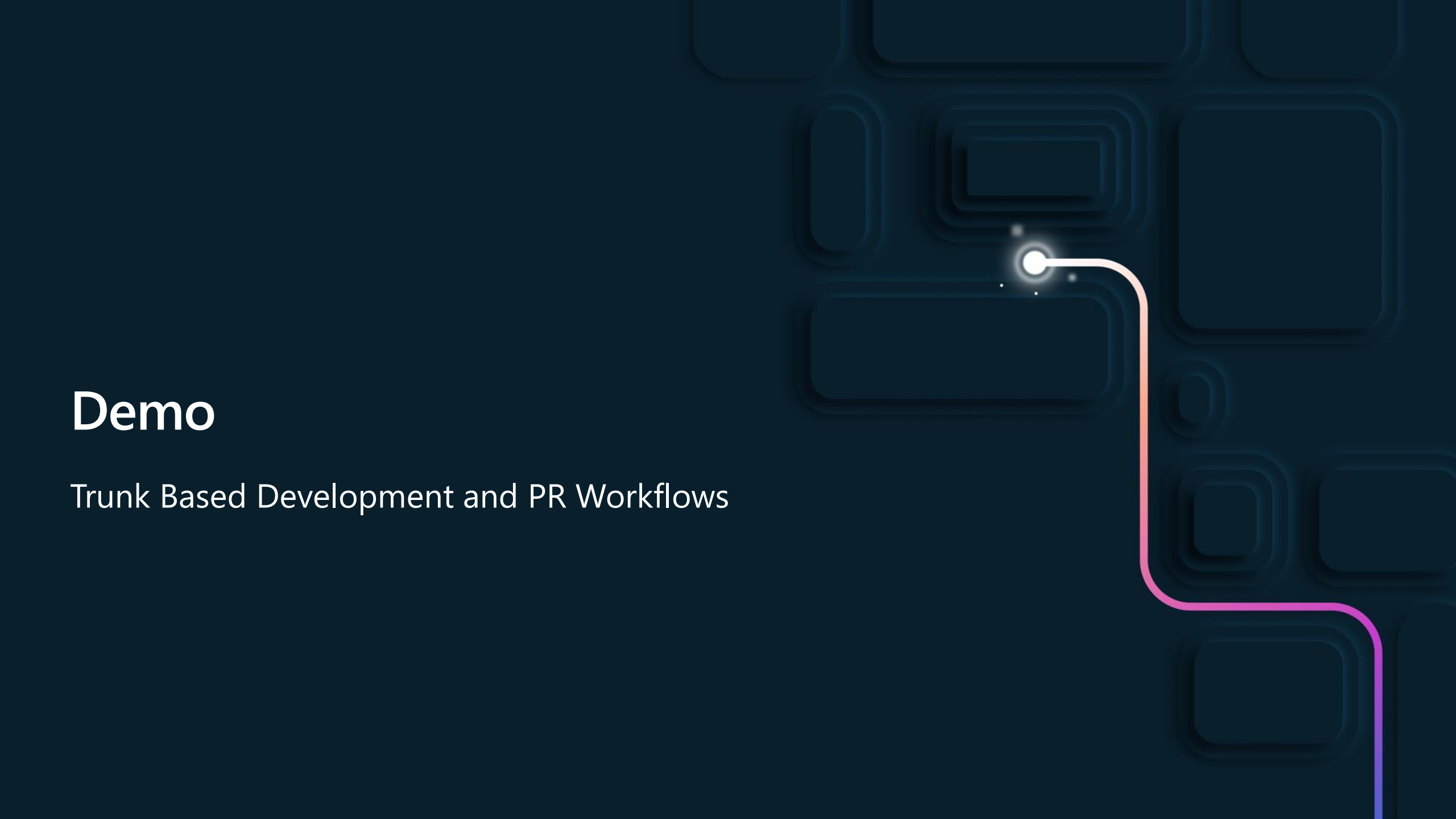
Short-lived feature branches

Trunk-based development - Releases



Demo

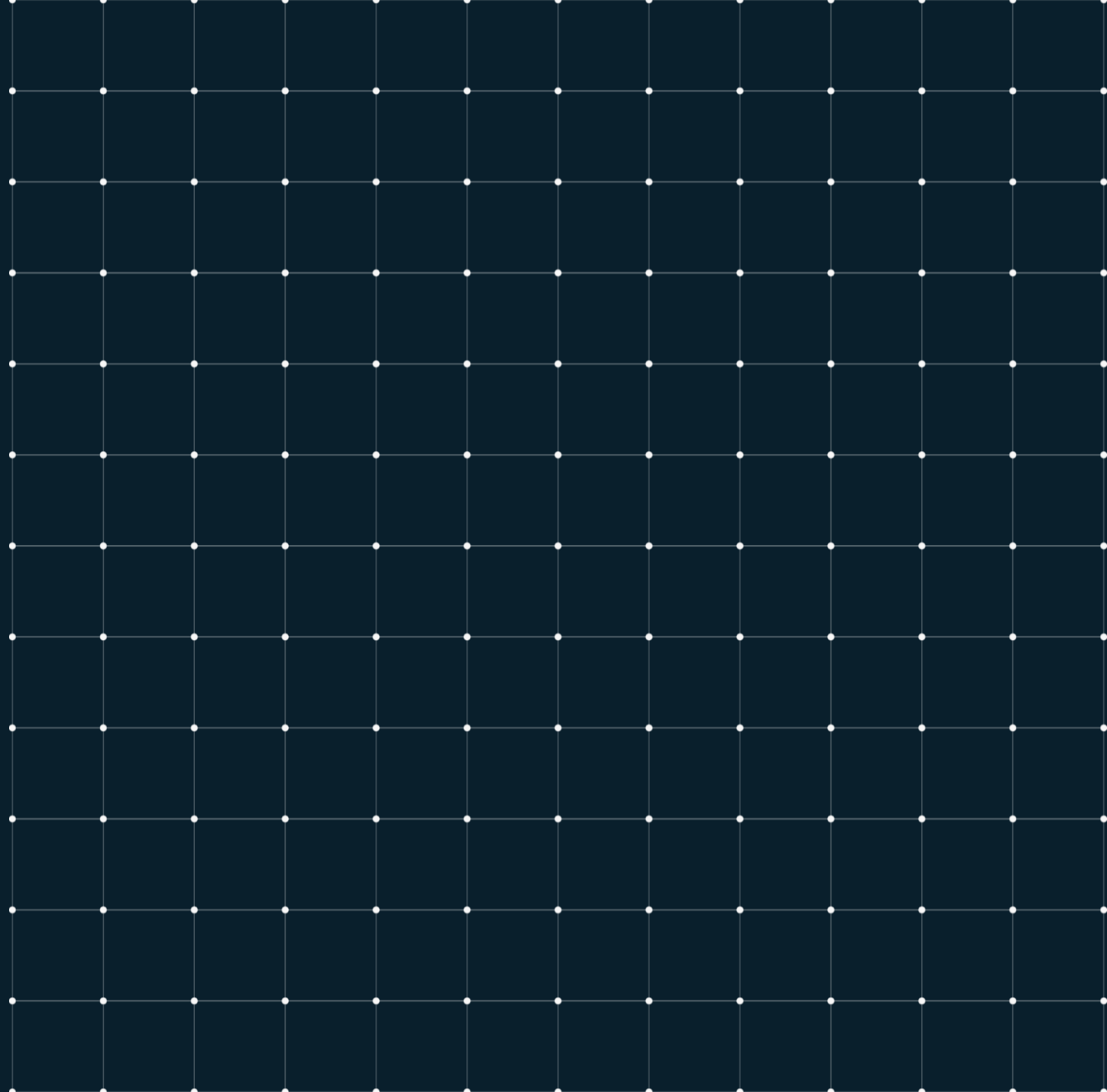
Trunk Based Development and PR Workflows



Pre-Production

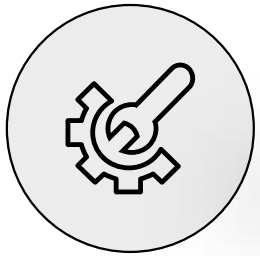
GitHub Actions

Azure App Service



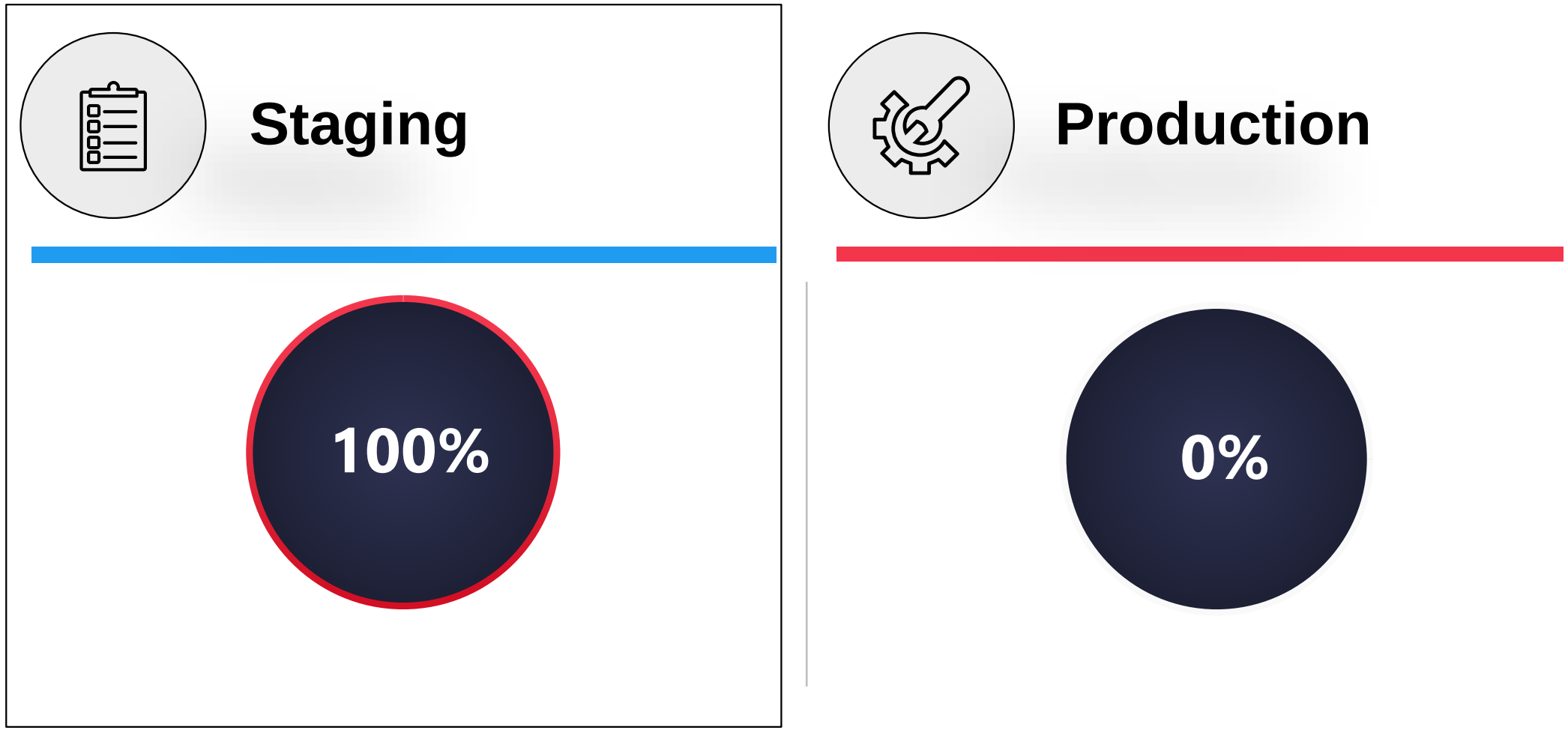
Deployment slots

Staging

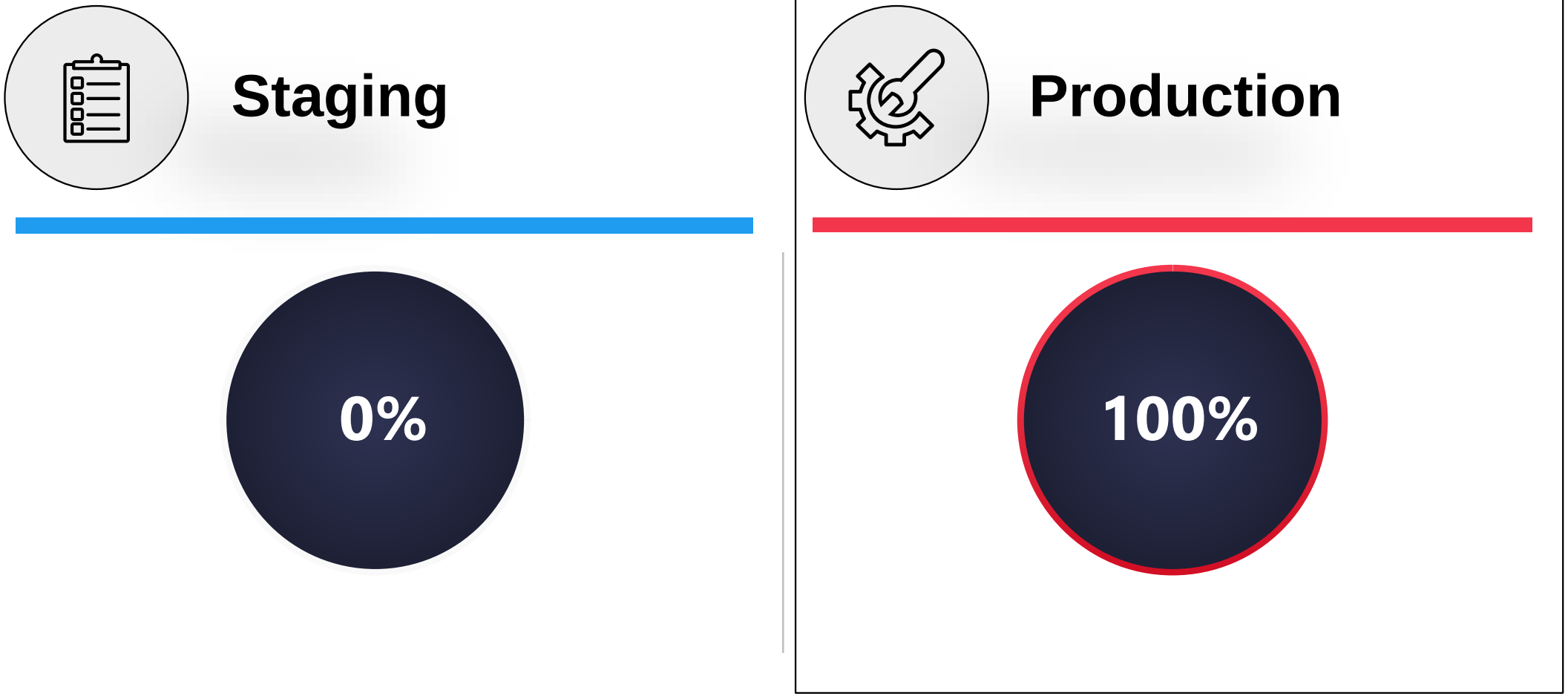


Production

Deployment slots – Staging traffic

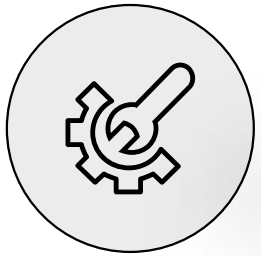


Deployment slots – Production traffic



Deployment slots – Canary deployment

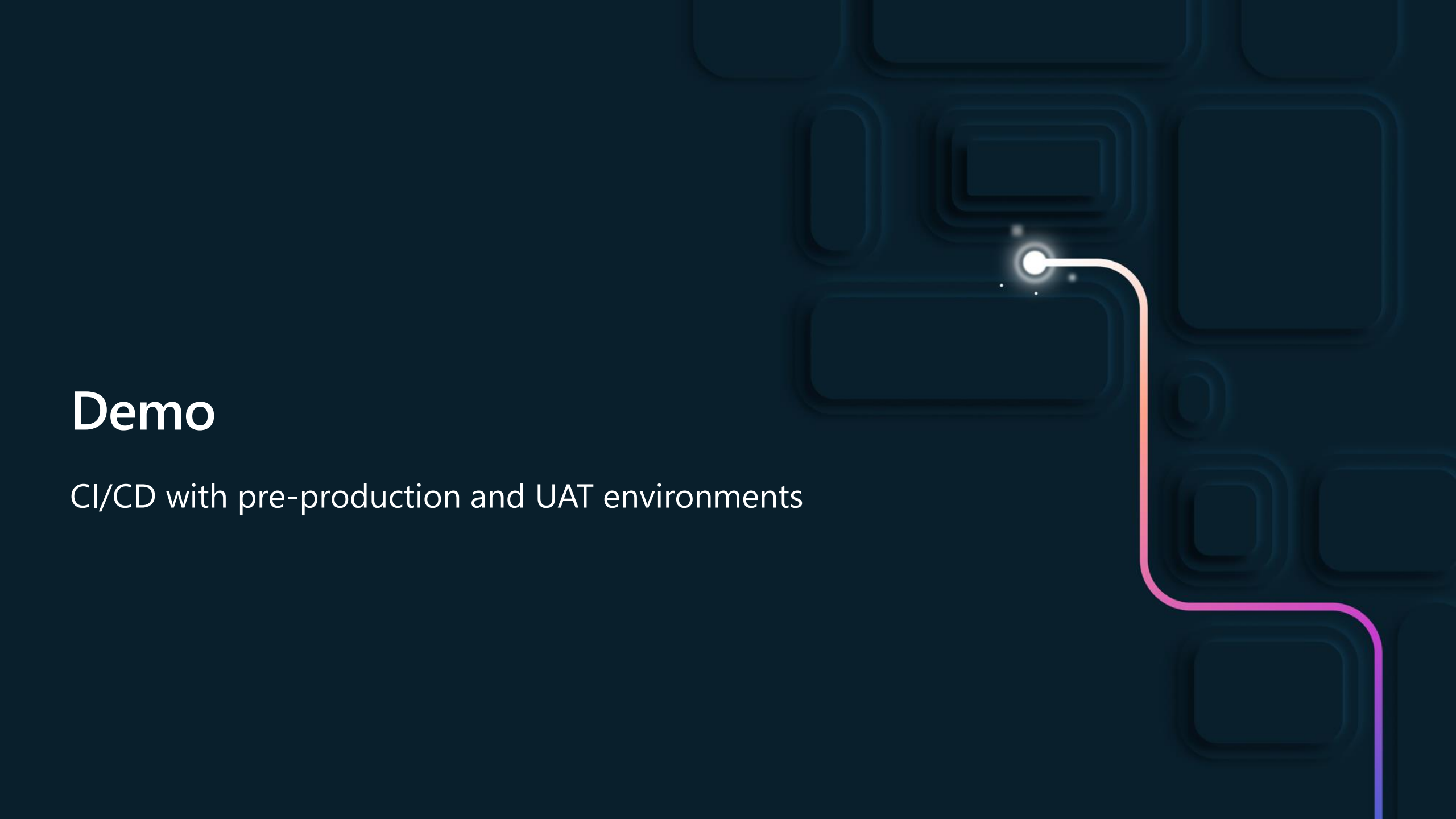
Staging



Production

Demo

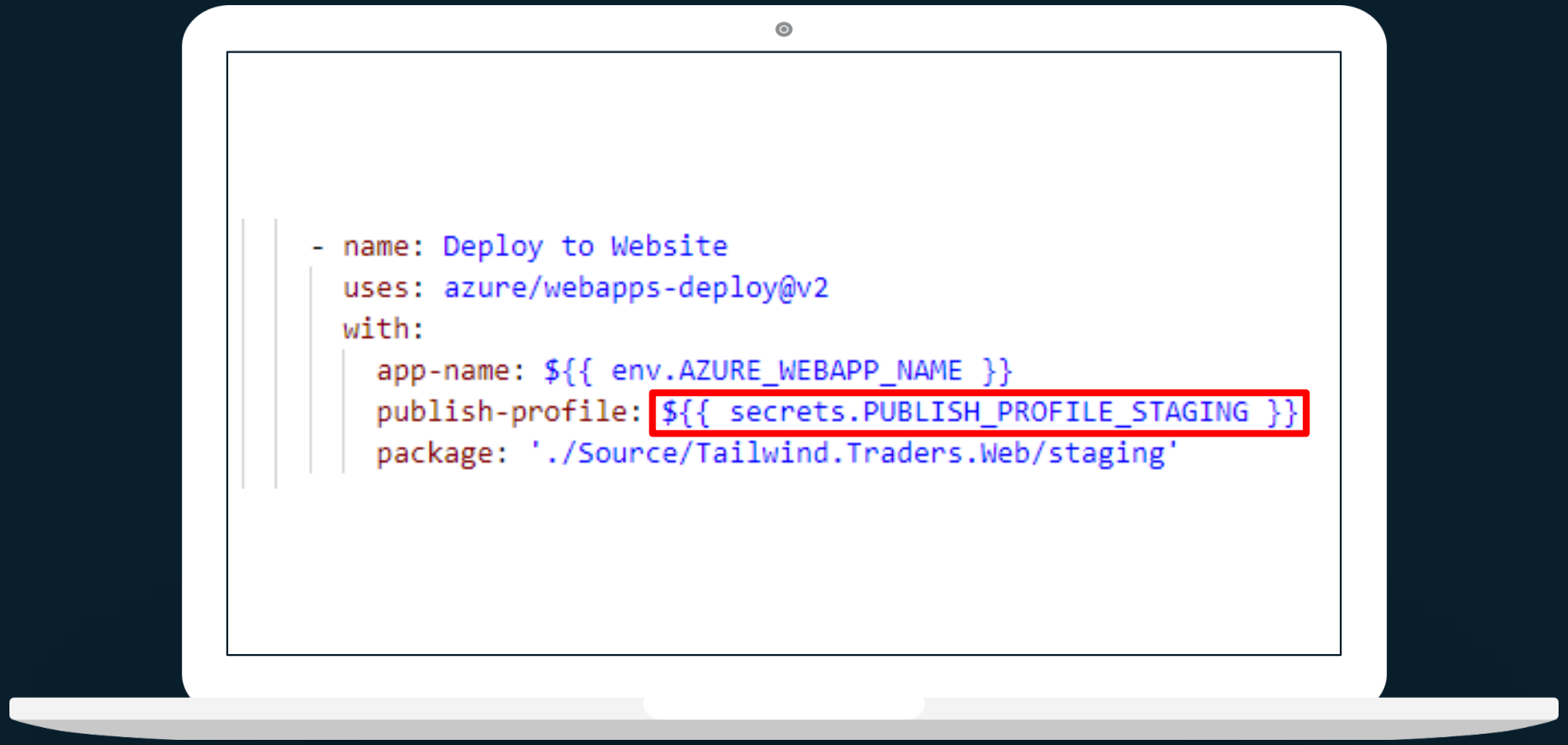
CI/CD with pre-production and UAT environments



Handling Keys and Credentials

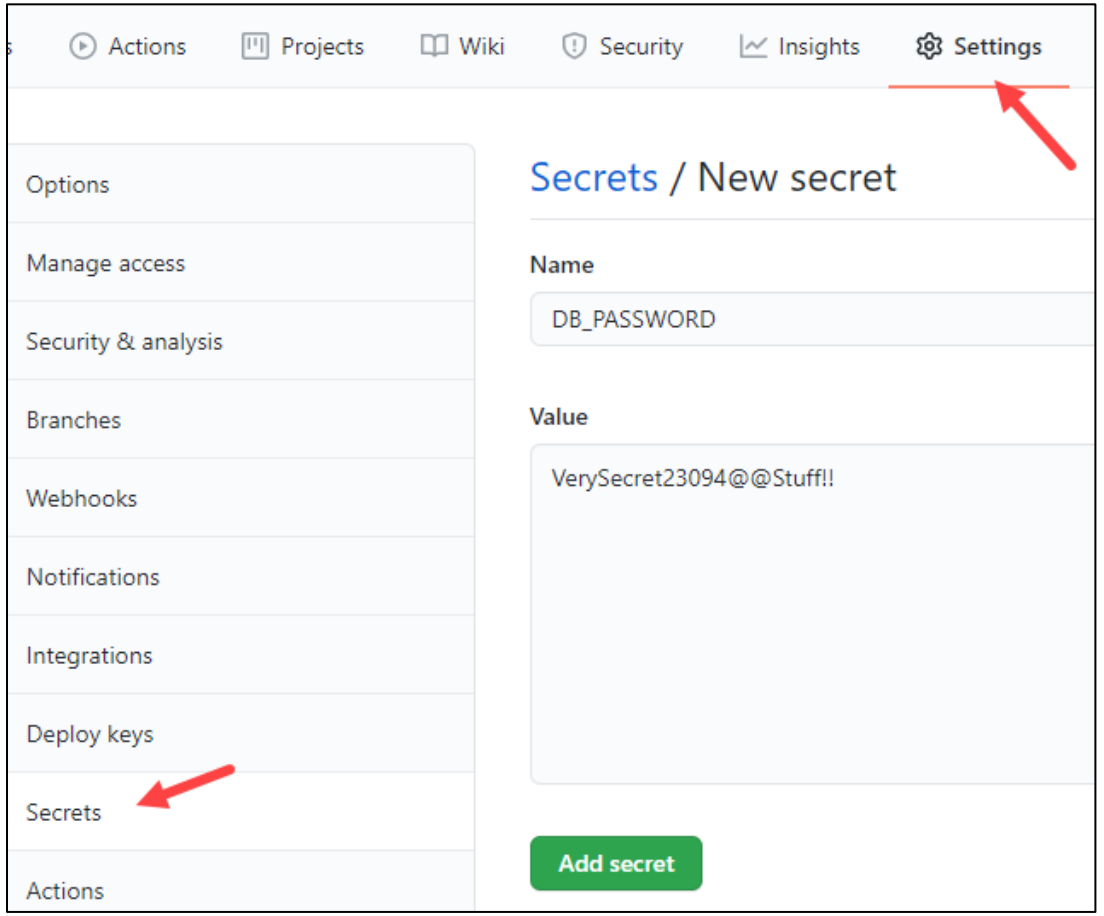
Keeping secrets secret

GitHub Secrets

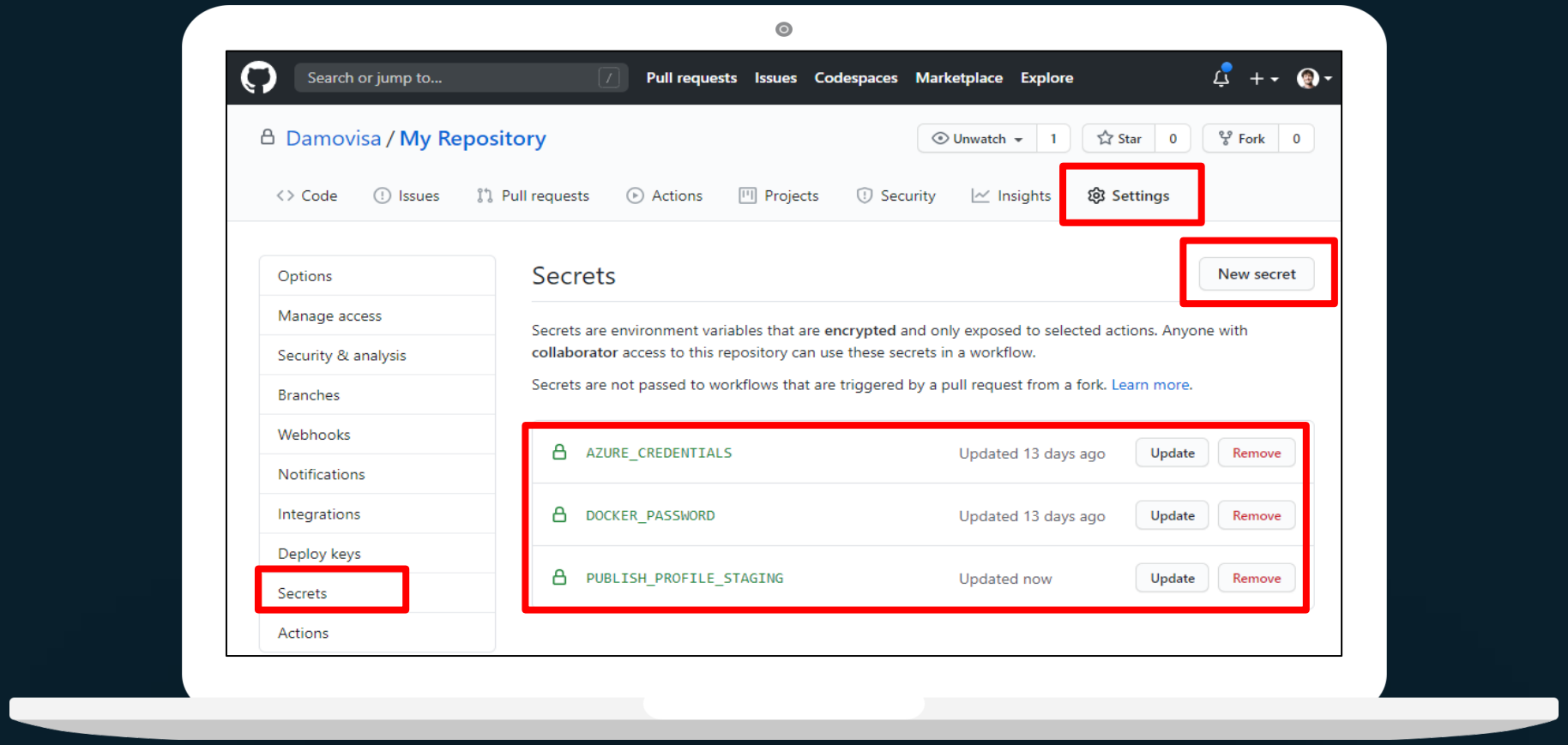


Create encrypted secrets

- Like environment variable but encrypted
- Created at repository or organization level
- Created/assigned in the GitHub UI



GitHub Secrets settings



Use secrets in a workflow

- Secrets not automatically passed to runners
- Can be passed as inputs or as environment variables
- Avoid passing secrets in command-line arguments

```yaml
steps:
  - name: Test Database Connectivity
    with:
      db_username: ${{ secrets.DBUserName }}
      db_password: ${{ secrets.DBPassword }}
```

```yaml
steps:
  - shell: pwsh
    env:
      DB_PASSWORD: ${{ secrets.DBPassword }}
    run: |
      db_test "$env:DB_PASSWORD"
```

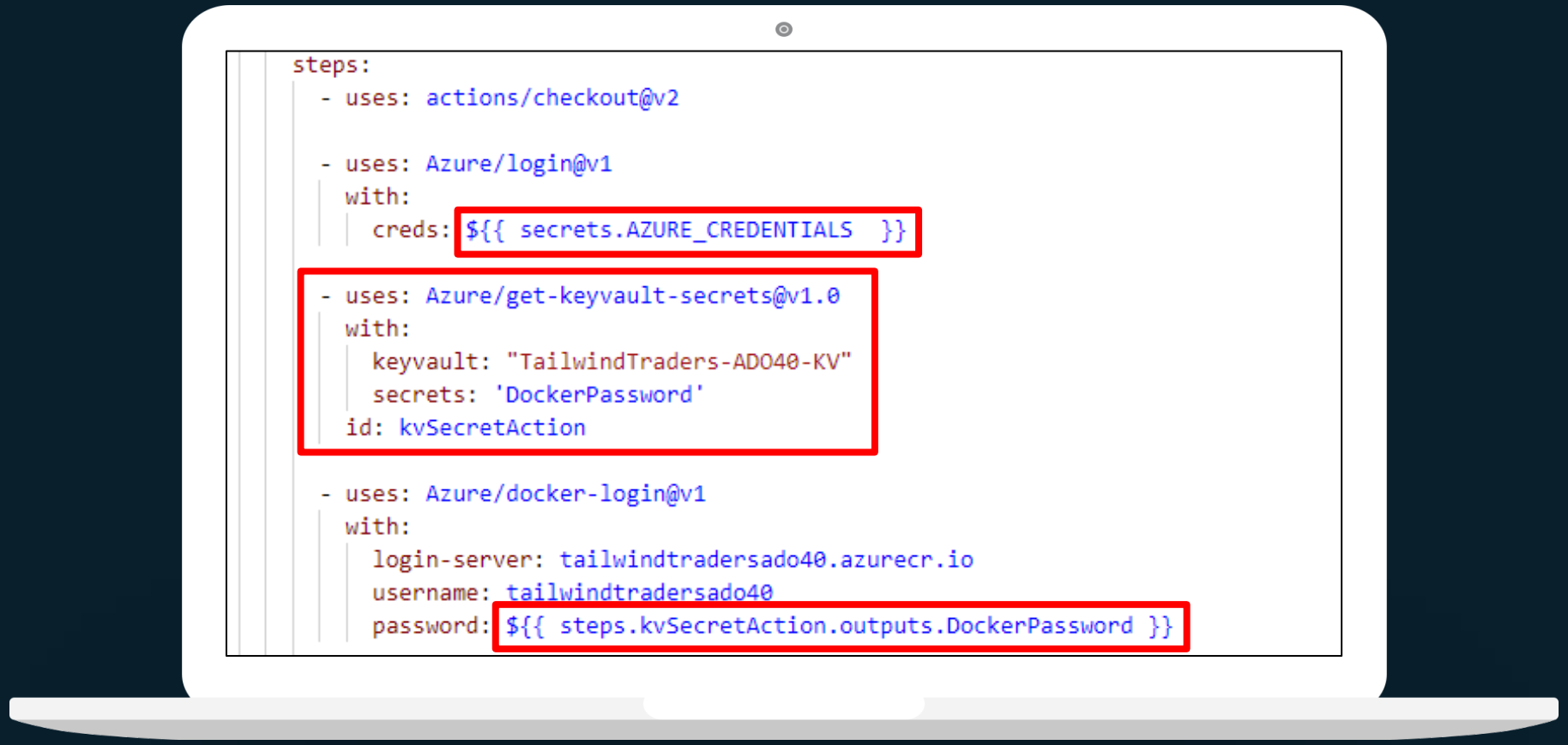
Azure Key Vault

Keys

Secrets

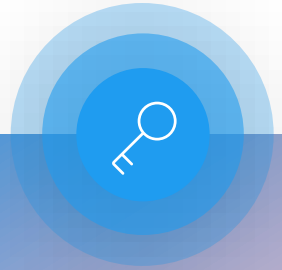
Certificates

Azure Key Vault actions

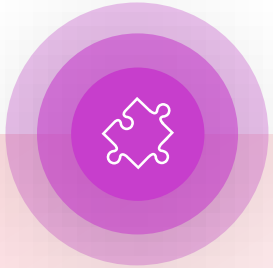


```yaml
steps:
  - uses: actions/checkout@v2

  - uses: Azure/login@v1
    with:
      creds: ${{ secrets.AZURE_CREDENTIALS  }}

  - uses: Azure/get-keyvault-secrets@v1.0
    with:
      keyvault: "TailwindTraders-ADO40-KV"
      secrets: 'DockerPassword'
    id: kvSecretAction

  - uses: Azure/docker-login@v1
    with:
      login-server: tailwindtradersado40.azurecr.io
      username: tailwindtradersado40
      password: ${{ steps.kvSecretAction.outputs.DockerPassword }}
```

Future ideas for Tailwind Traders



Rotate credentials in Key Vault when an admin asks a Teams Bot



Automatically build a new dev environment and fork a repository based on a properly-formatted issue

Pull commit messages since last production release and build release notes for customers

Summary

Add CD to your pipeline!

Continuously deploy somewhere

Protect production with GitHub Environments

Centralize Secrets Storage

Microsoft

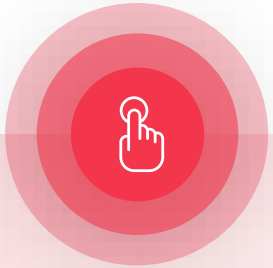
Delivering Changes to the Cloud

DevOps Learning Path

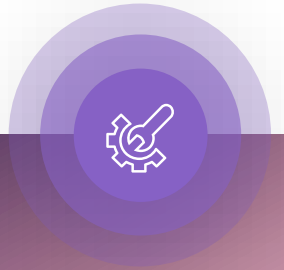
- Getting Started with DevOps
- Managing the Flow of Work
- Shift security “left” in your CI/CD process
- **Delivering changes to cloud**
- Performance Monitoring and maintenance

Goals for this Session

Continuous Integration
Continuous Delivery /
Deployment

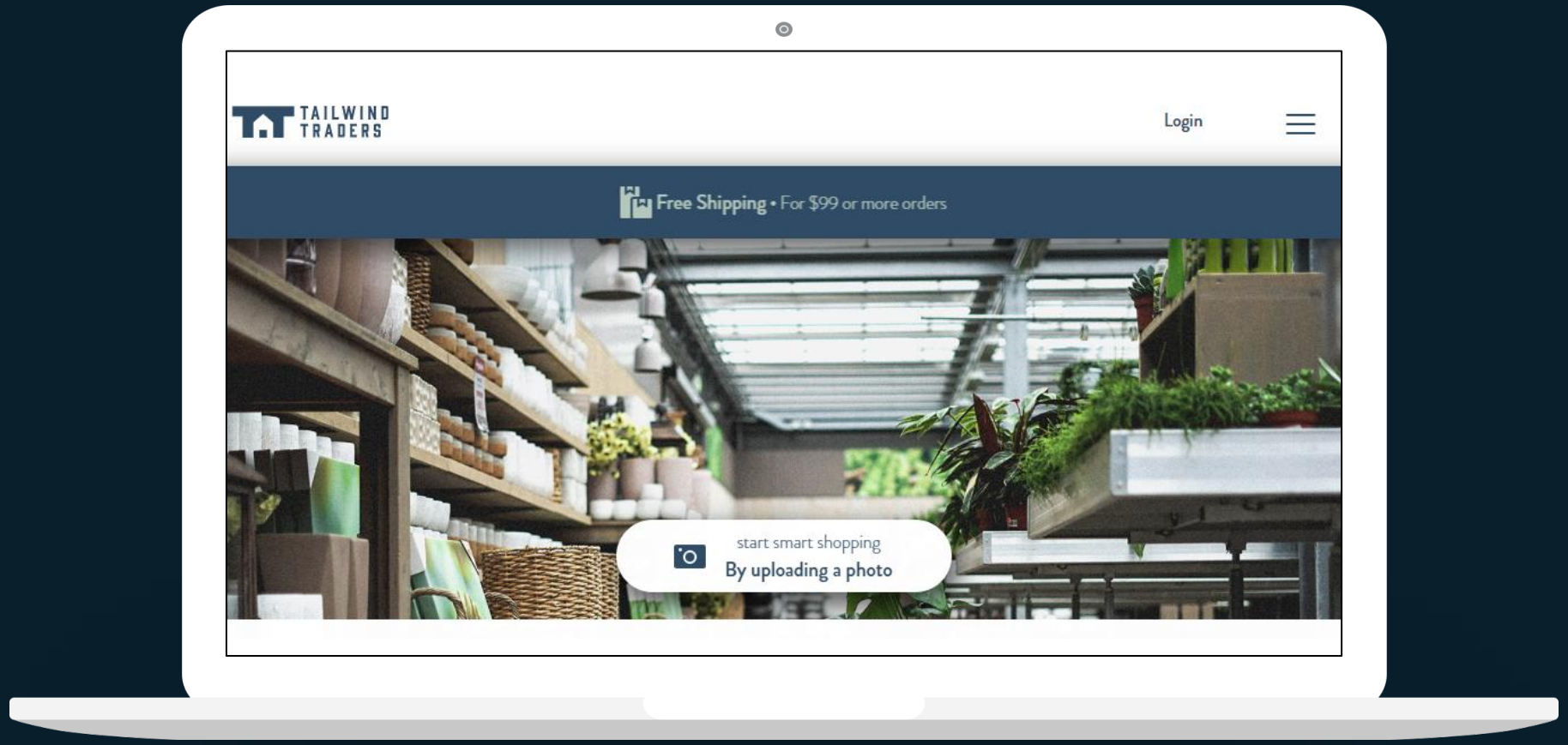


Trunk Based
Development



Protect Production /
Secrets

Tailwind Traders

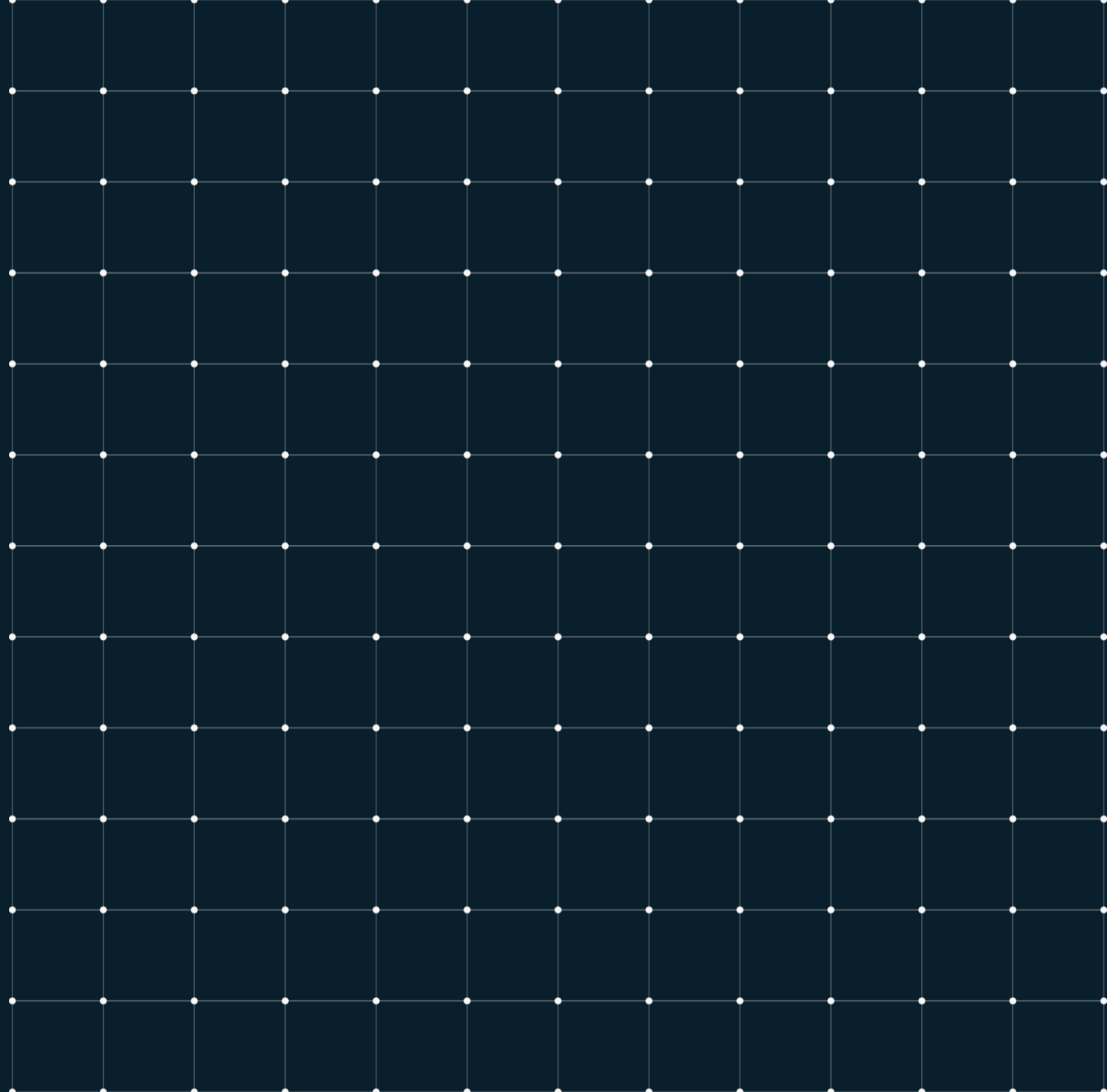


CI and CD

Continuous Integration

Continuous Delivery

Continuous Deployment



Continuous Integration

Your changes work with everyone else's changes

Your code still builds

Your tests still run

Continuous Delivery

You have a deployable piece of work

Including infrastructure and dependencies

Deploy from build to a testing, staging, and/or production environment

Continuous Deployment

Deploy that piece of work

Doesn’t have to be to Production

Trustworthy and reproducible

CI and CD

Continuous Integration –
Develop phase. Build, test, and validate code.

Continuous Delivery –

Automates delivery. New build artifact is available, artifact is deployed.

Continuous Deployment –

From when you commit and check in code to production, everything is automated.

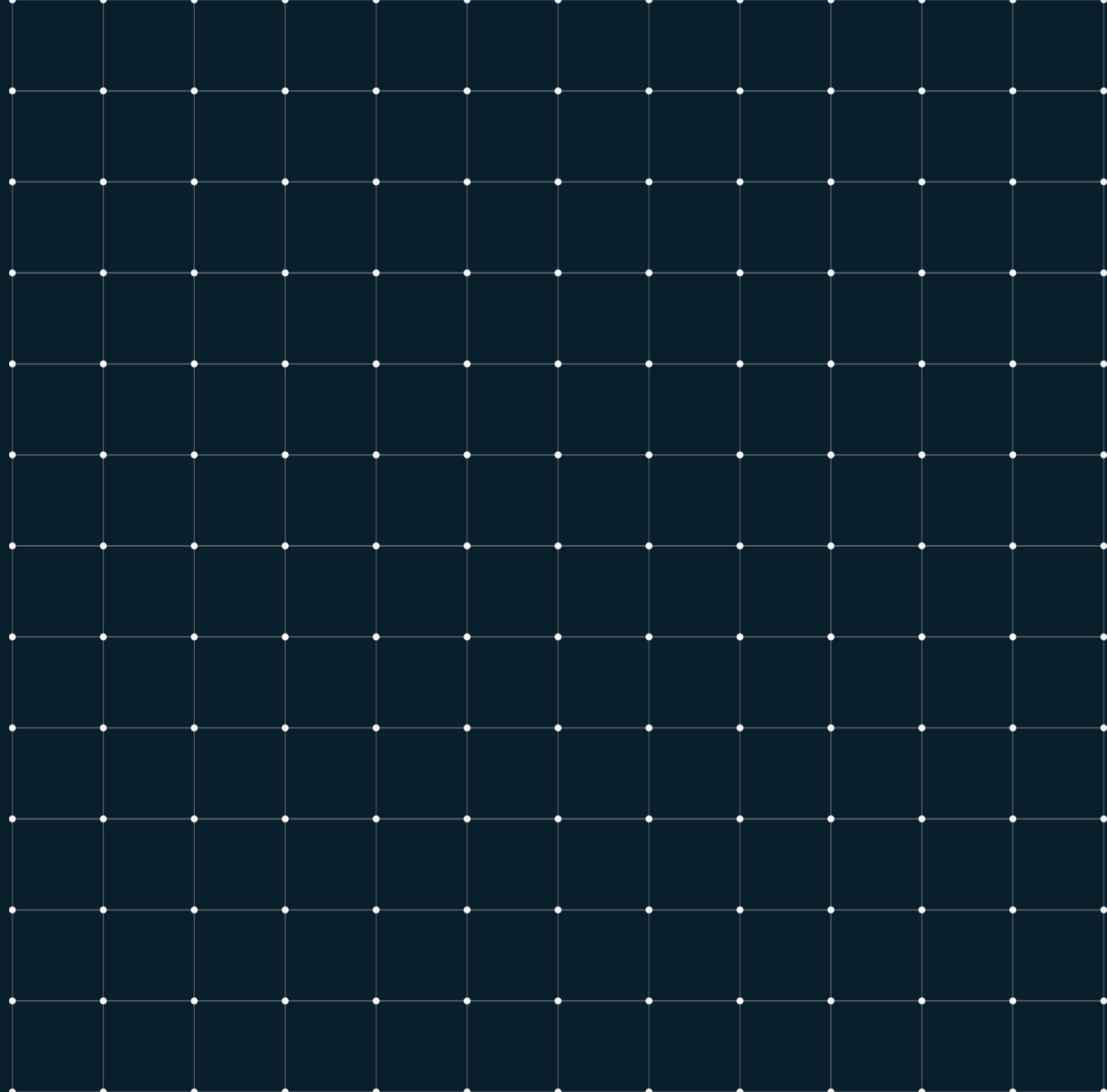
Protip

Always have Continuous Deployment to *somewhere*.

Don’t assume this version will deploy as cleanly as the last.

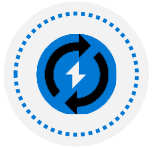
GitHub Actions

CI/CD with GitHub Actions

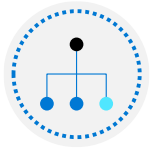


What are Actions?

Automations within the GitHub environment



Often used to build CI/CD implementations



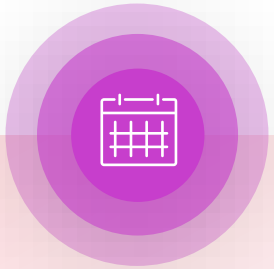
Based on YAML files living within GitHub repositories

Executed on GitHub or self-hosted runners

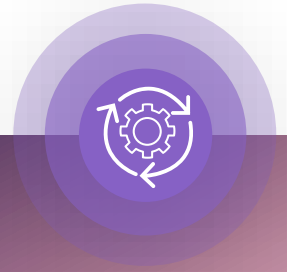
Large number of existing actions in the GitHub Marketplace

GitHub Actions

Automation for any software workflow



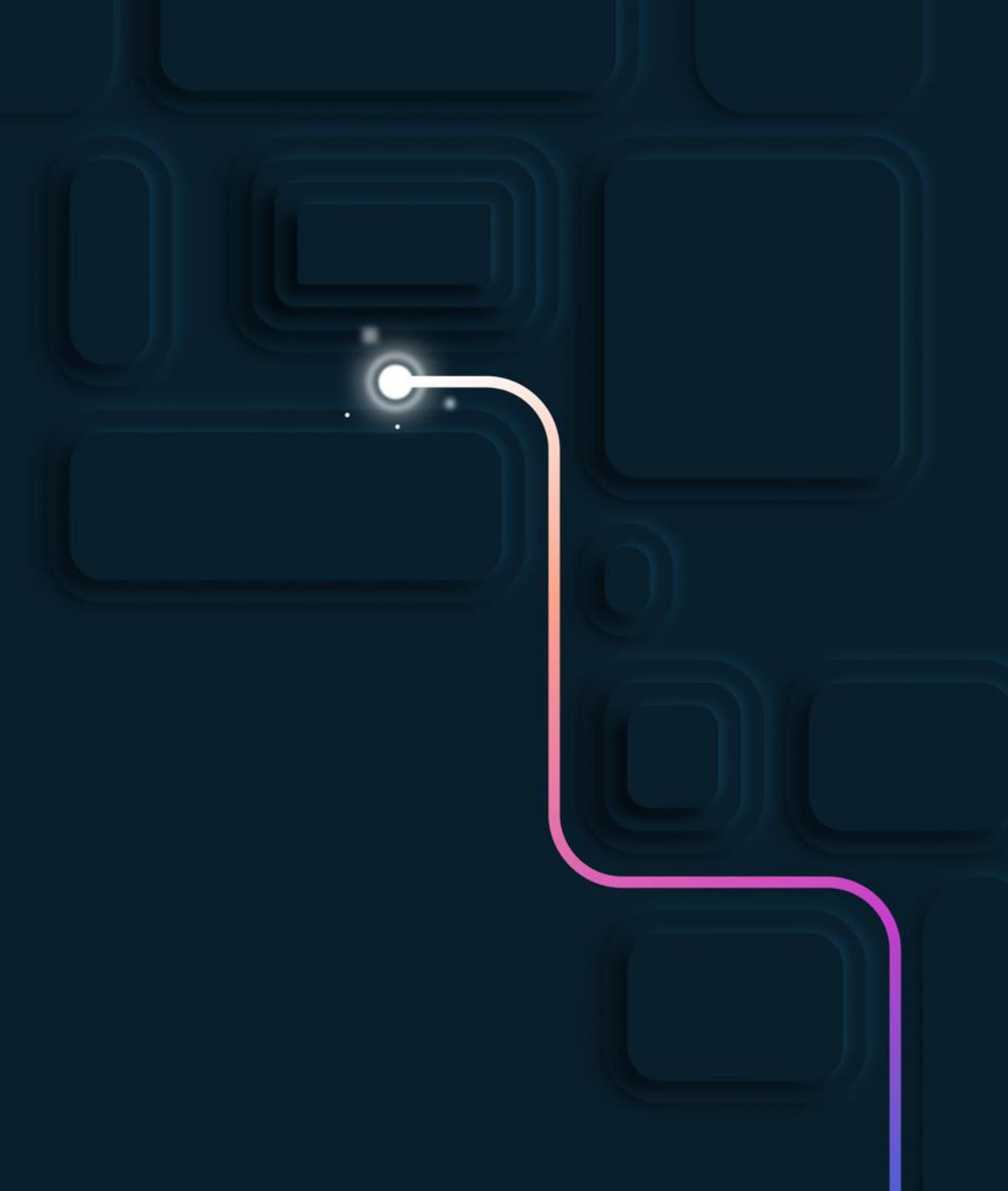
Dozens of events that can trigger workflows



Continuous Integration and Continuous Delivery

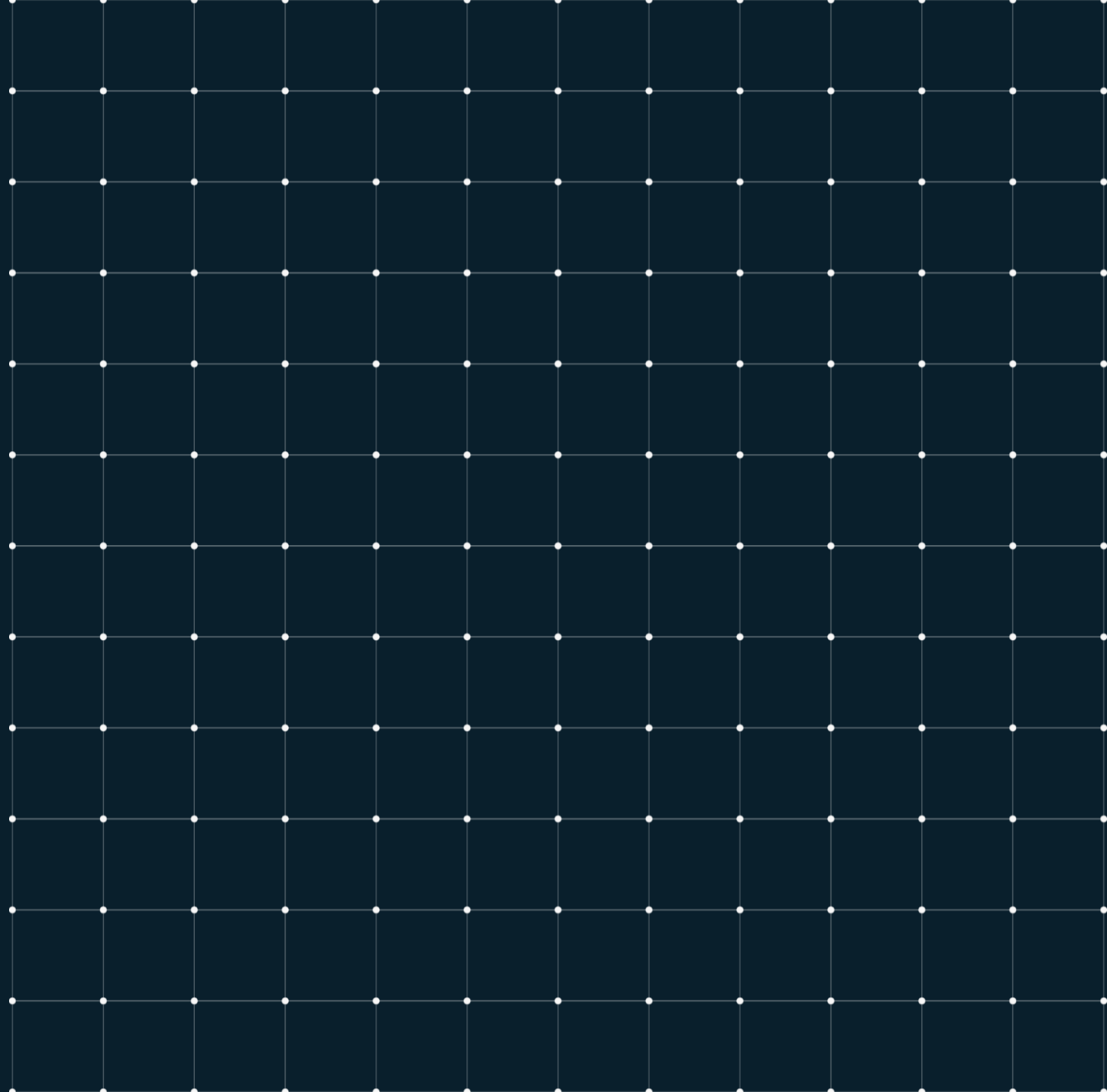
Demo

GitHub Actions – CI/CD

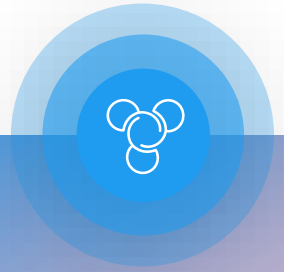


Protecting Production

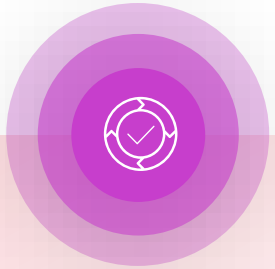
We might not want every change to go to Production



Pull Request Workflows



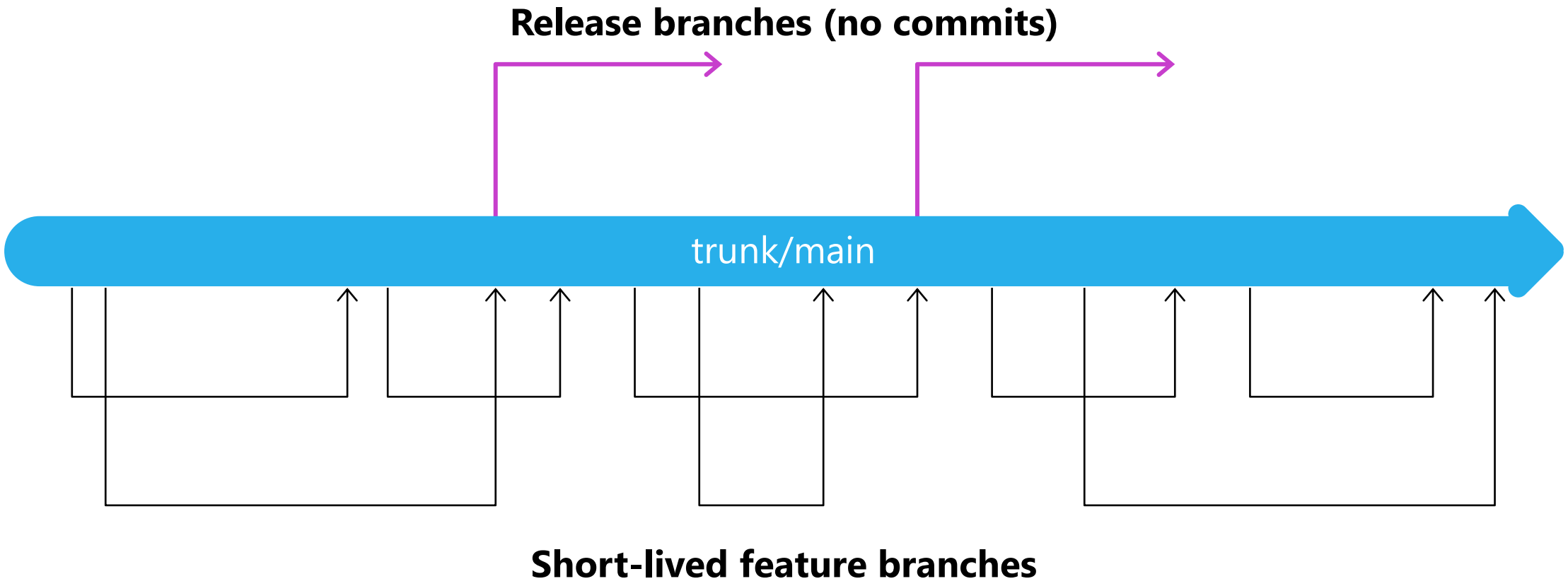
Workflow triggers on PR



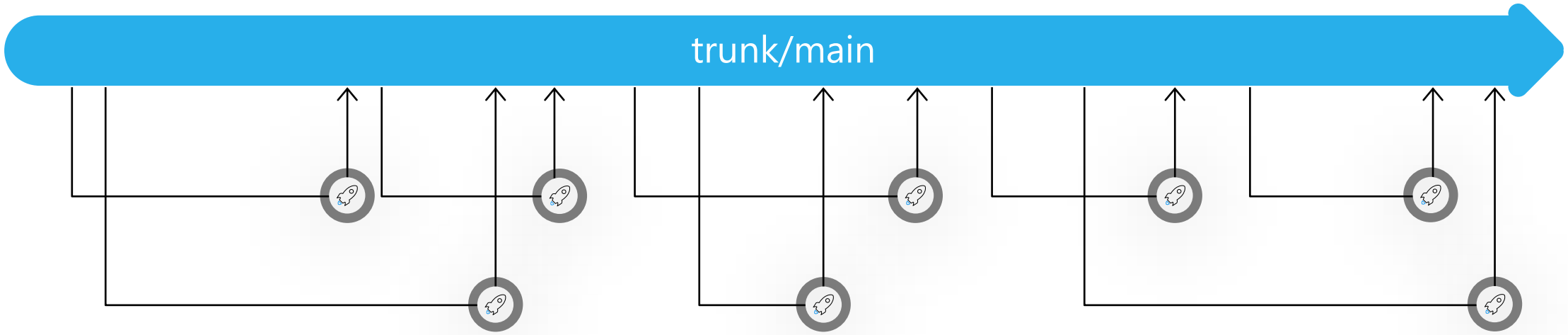
Build, test, deploy

Checks before merging to main branch

Trunk-based development

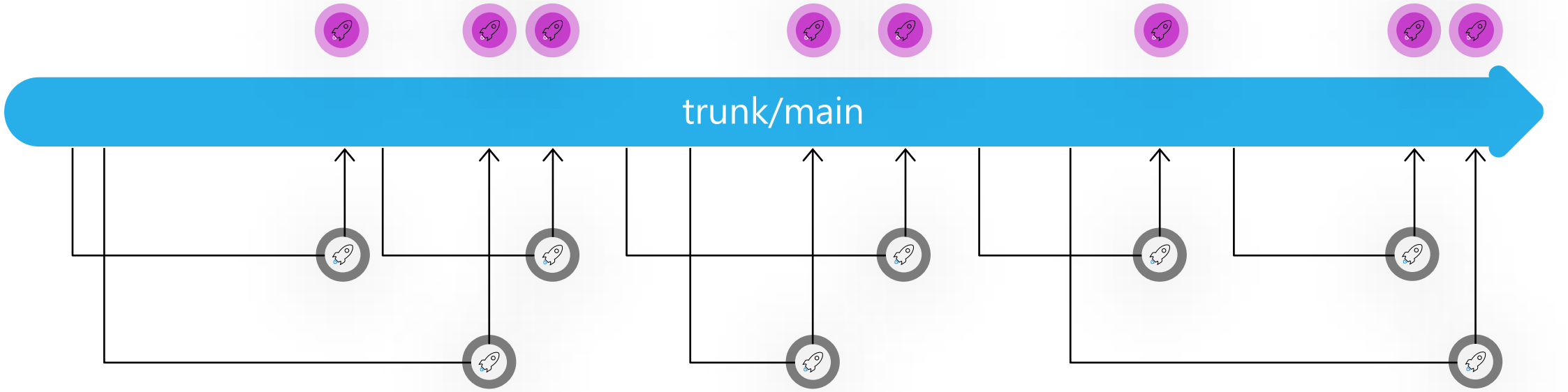


Trunk-based development – Topic Branches



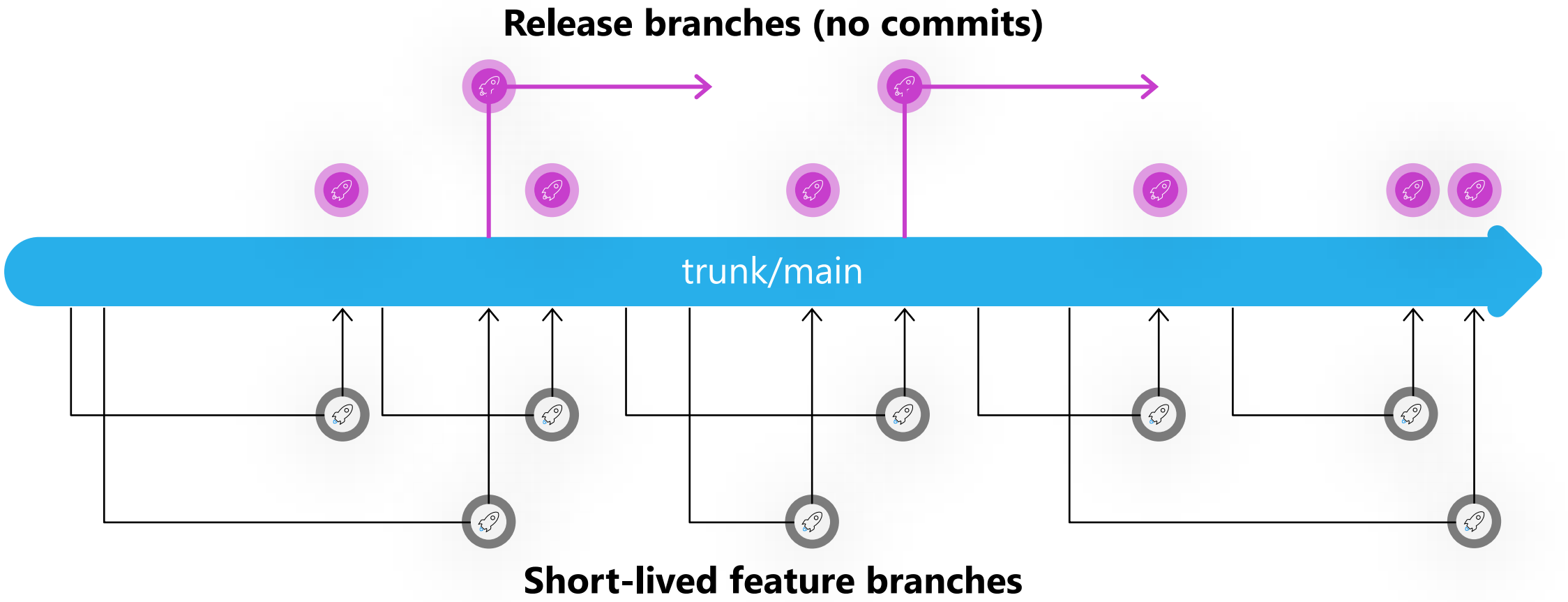
Short-lived feature branches

Trunk-based development – Merge, Build and Deploy



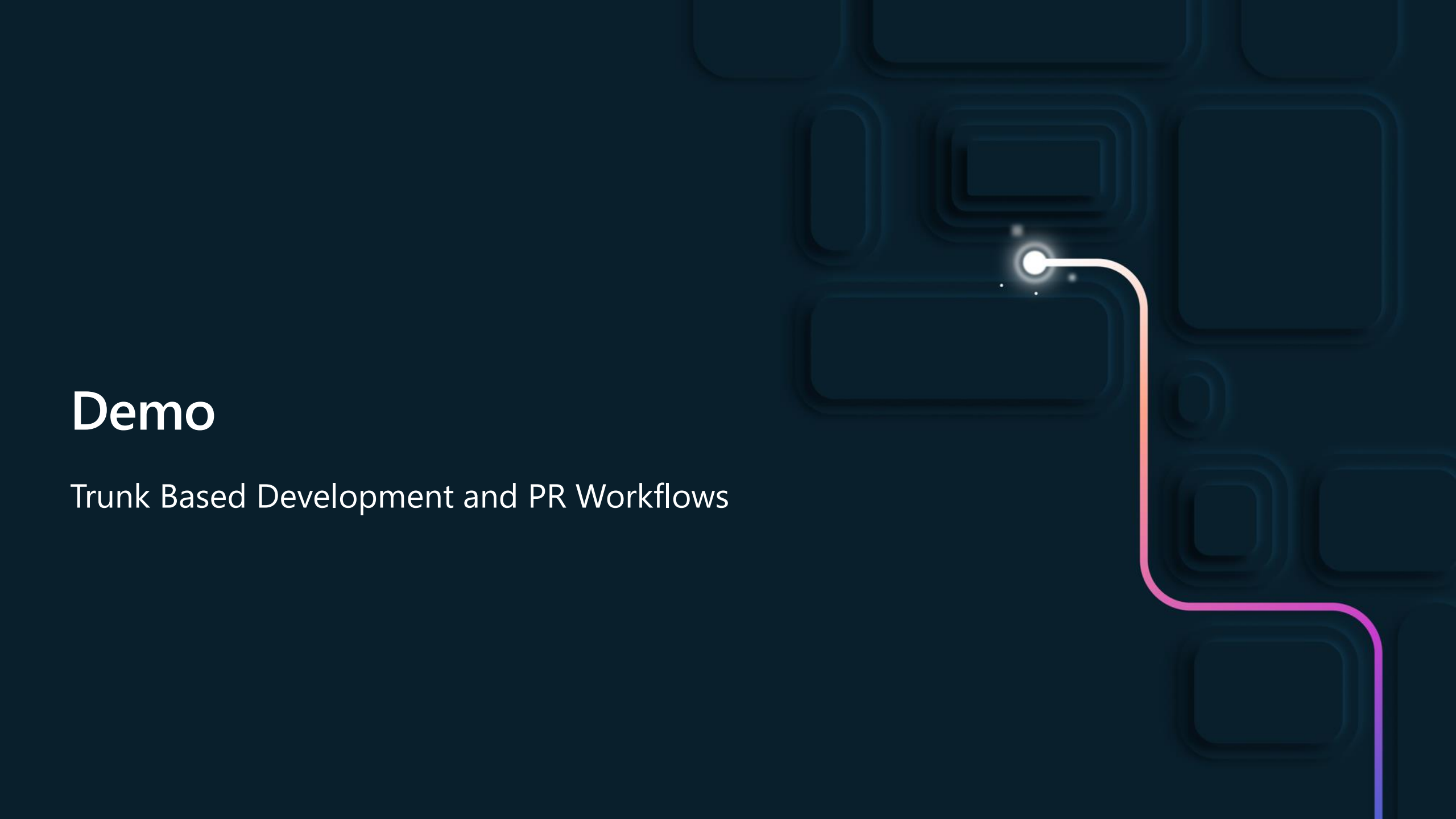
Short-lived feature branches

Trunk-based development - Releases



Demo

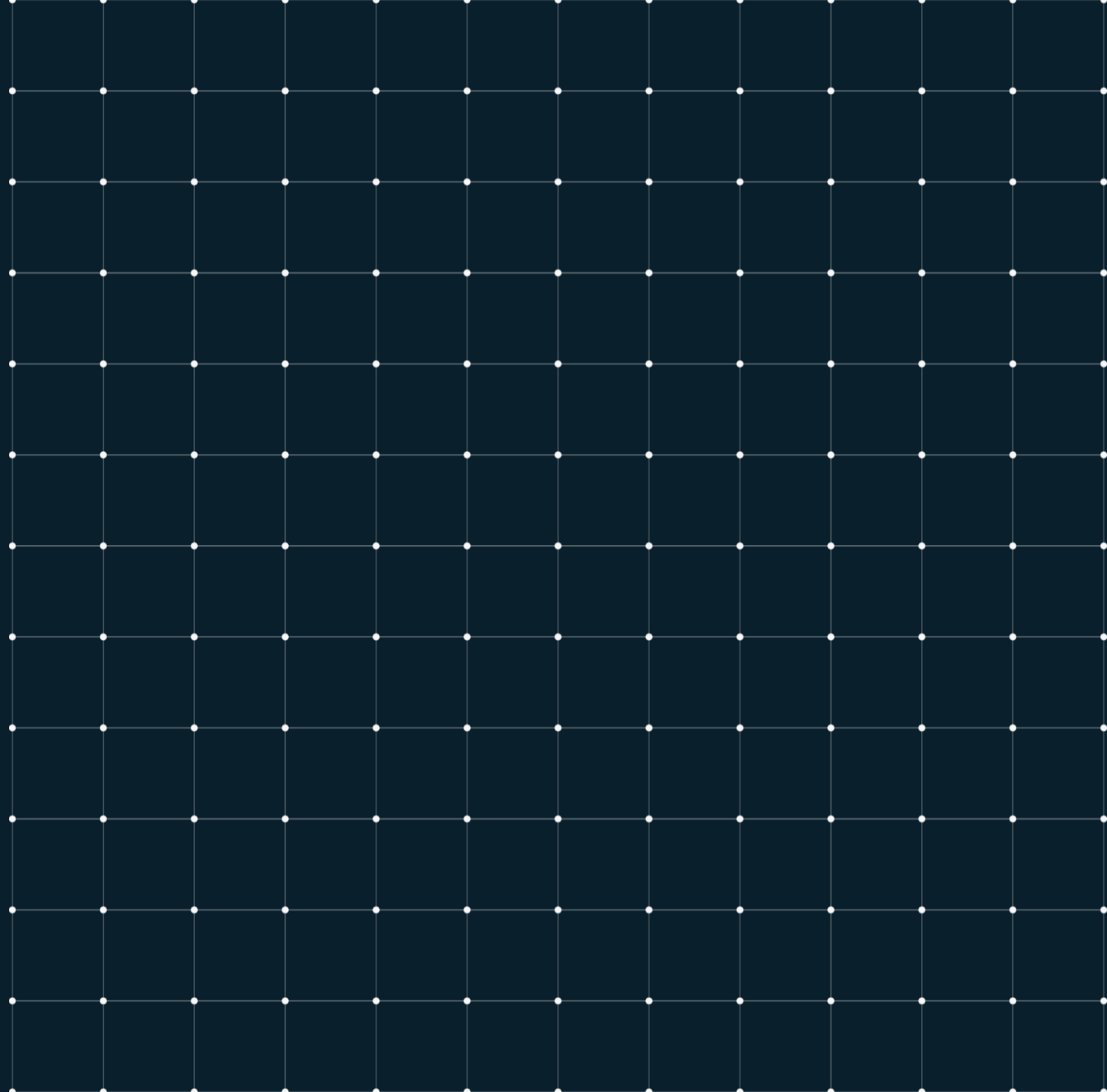
Trunk Based Development and PR Workflows



Pre-Production

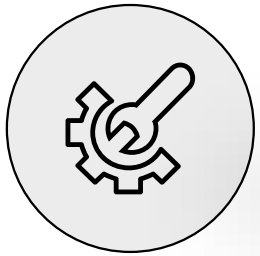
GitHub Actions

Azure App Service



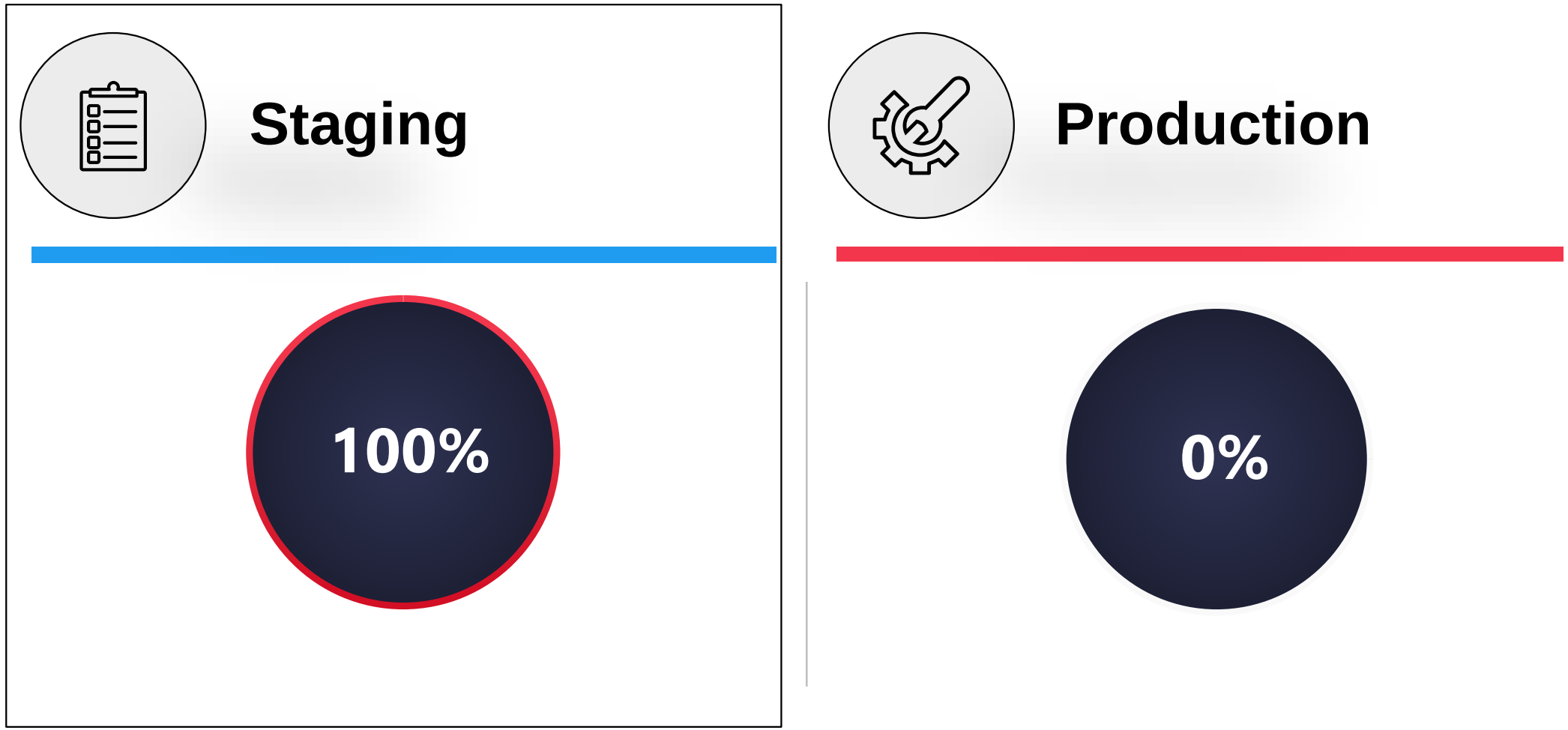
Deployment slots

Staging

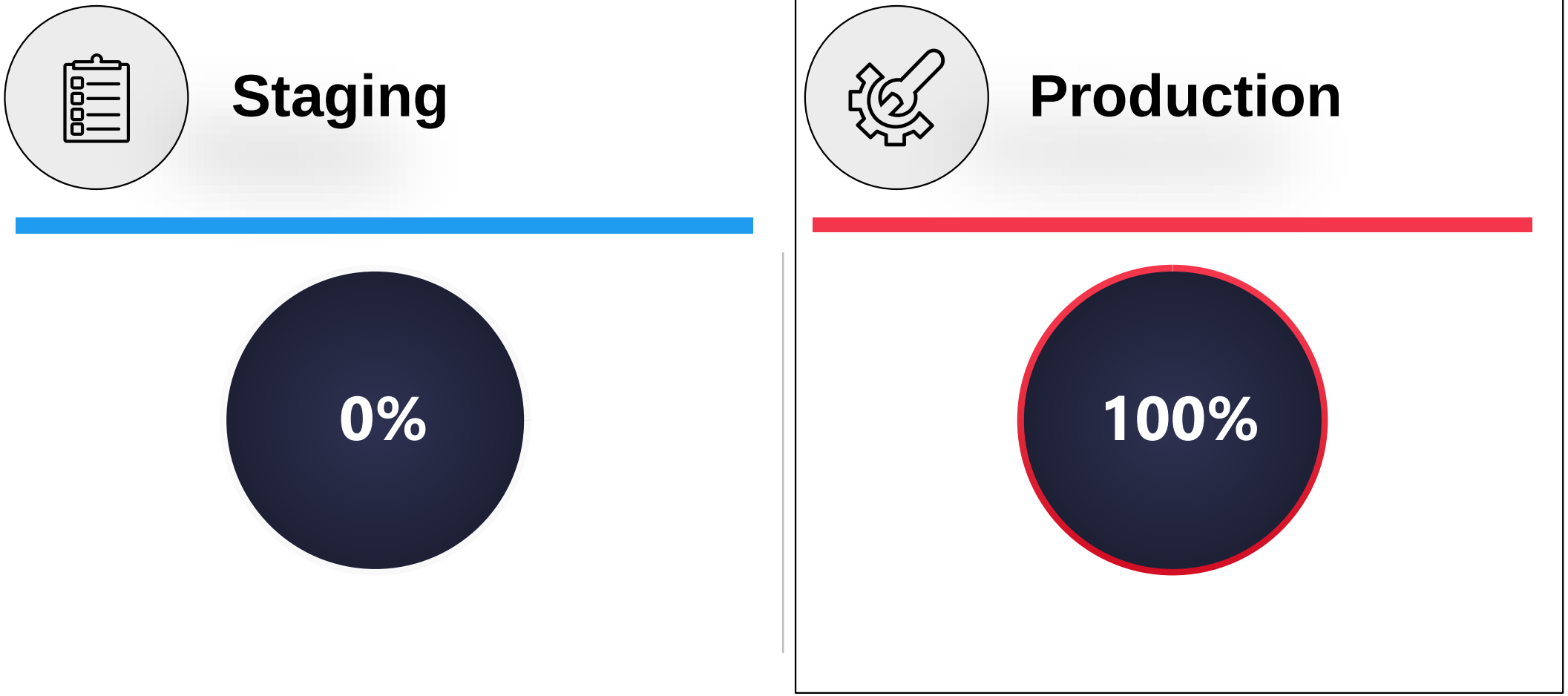


Production

Deployment slots – Staging traffic

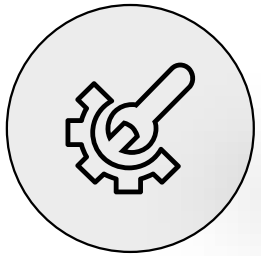


Deployment slots – Production traffic



Deployment slots – Canary deployment

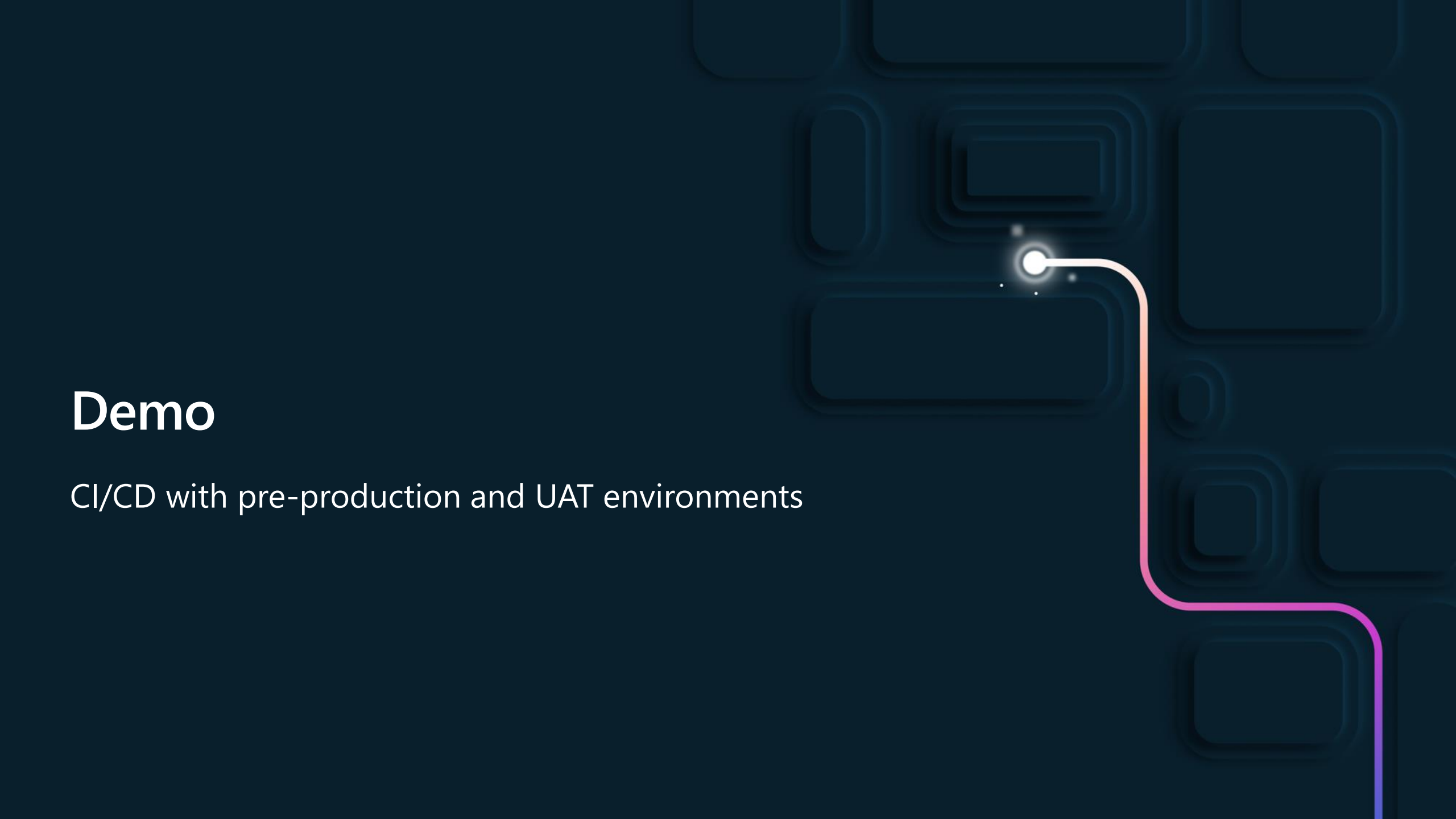
Staging



Production

Demo

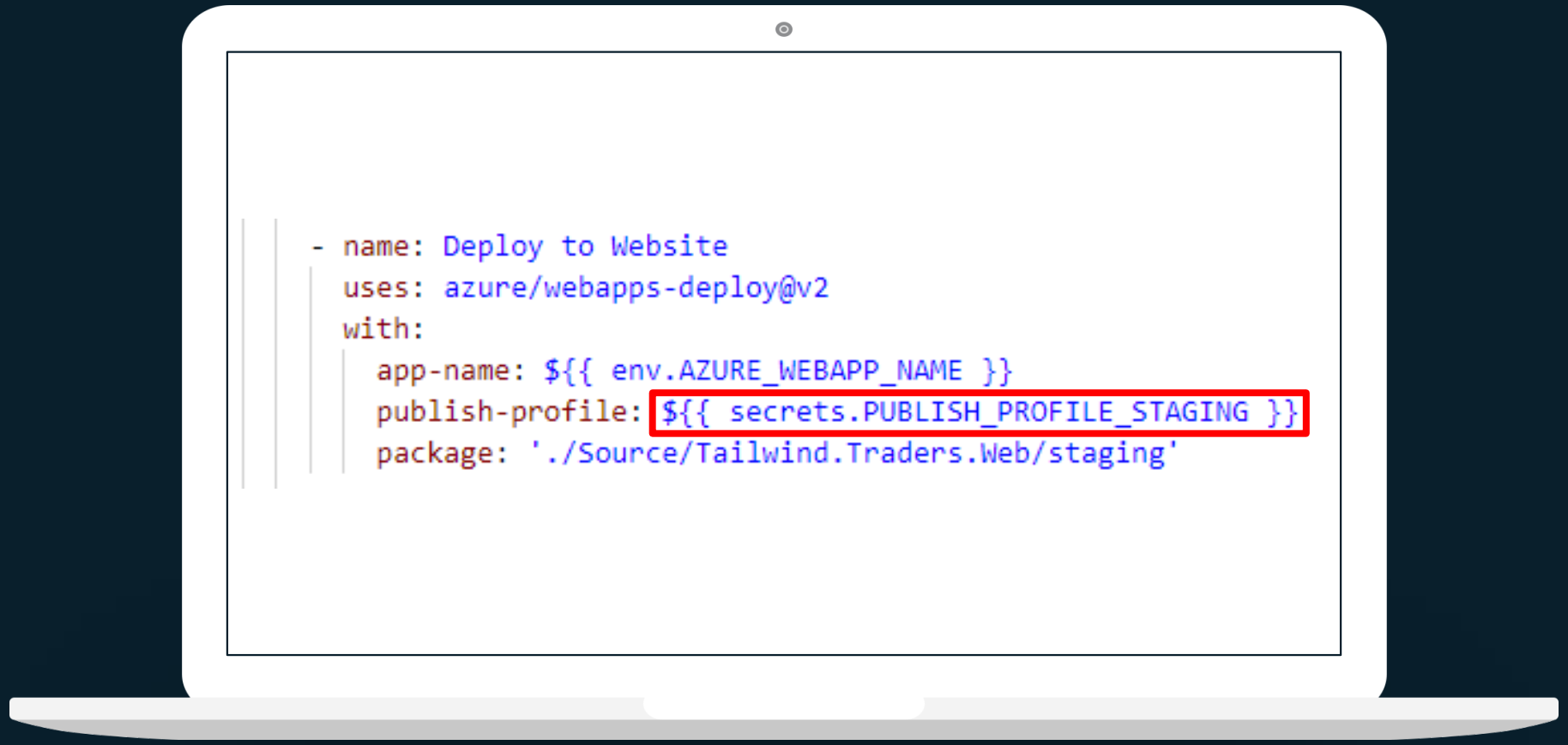
CI/CD with pre-production and UAT environments



Handling Keys and Credentials

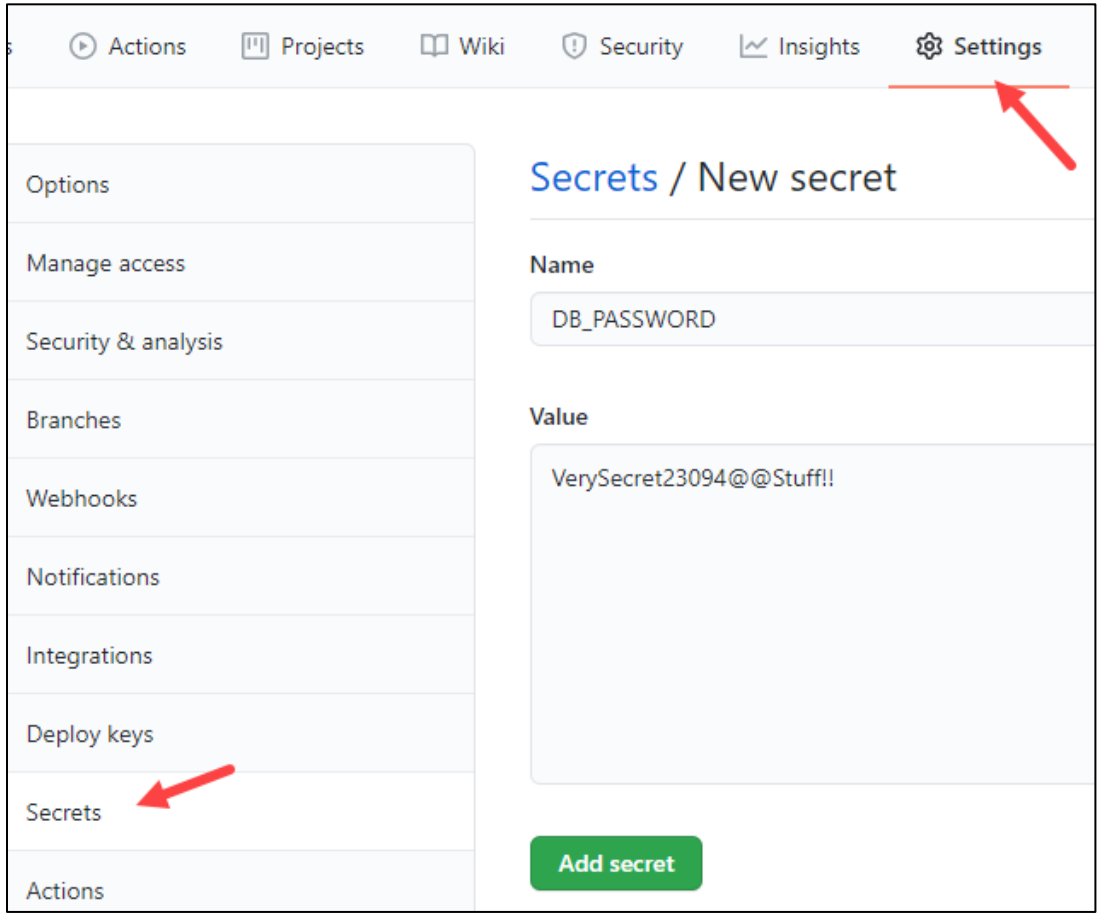
Keeping secrets secret

GitHub Secrets

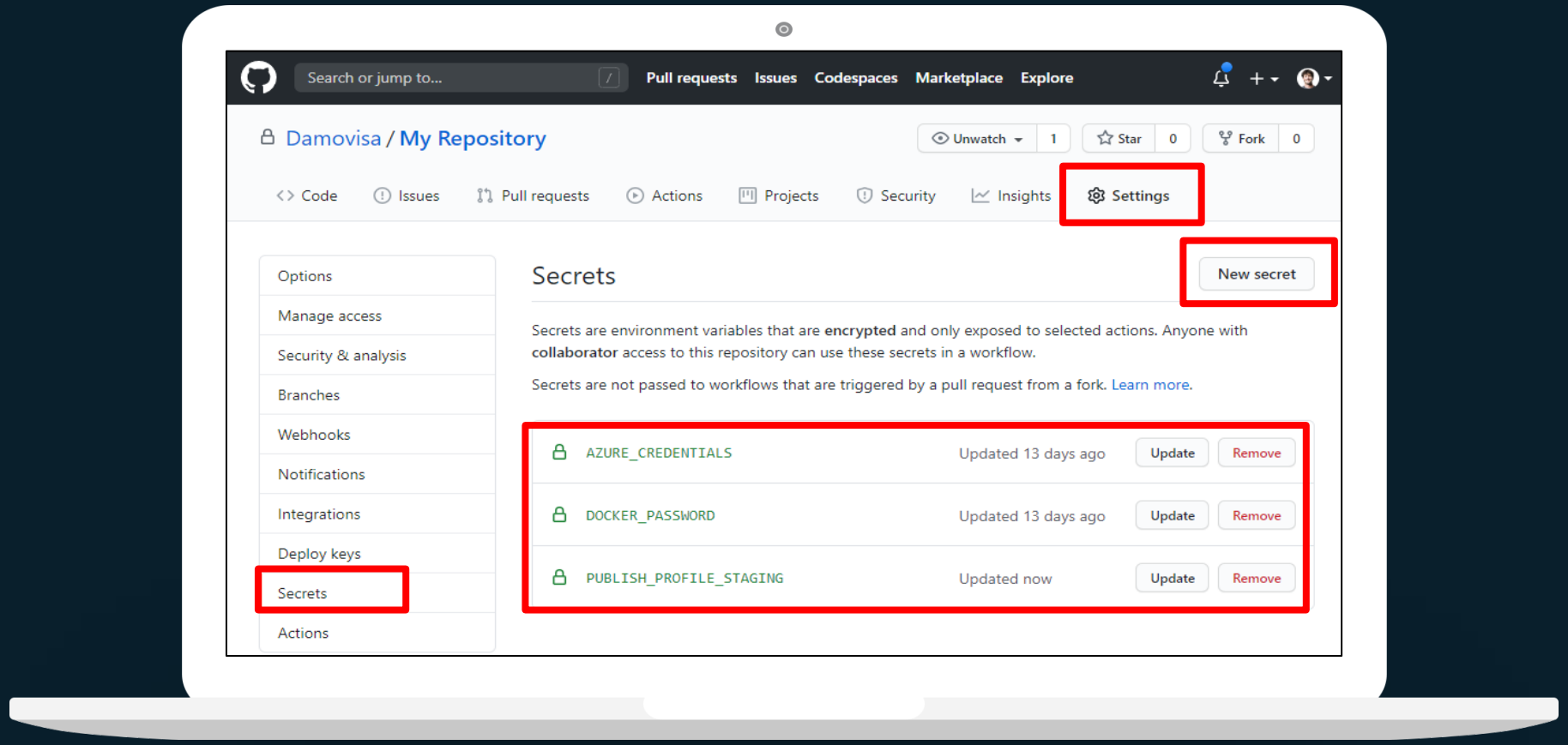


Create encrypted secrets

- Like environment variable but encrypted
- Created at repository or organization level
- Created/assigned in the GitHub UI



GitHub Secrets settings



Use secrets in a workflow

- Secrets not automatically passed to runners
- Can be passed as inputs or as environment variables
- Avoid passing secrets in command-line arguments

```yaml
steps:
  - name: Test Database Connectivity
    with:
      db_username: ${{ secrets.DBUserName }}
      db_password: ${{ secrets.DBPassword }}
```

```yaml
steps:
  - shell: pwsh
    env:
      DB_PASSWORD: ${{ secrets.DBPassword }}
    run: |
      db_test "$env:DB_PASSWORD"
```

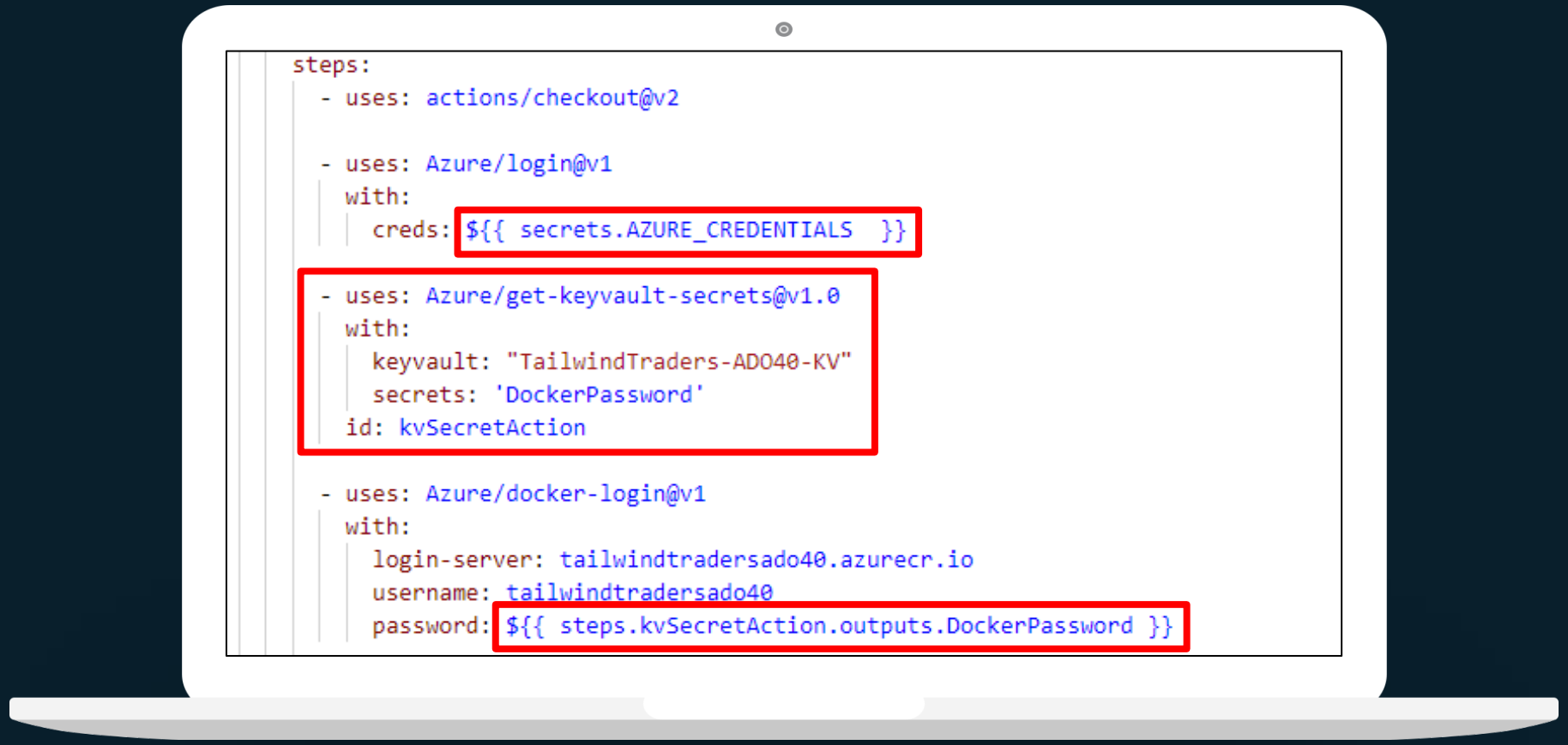
Azure Key Vault

Keys

Secrets

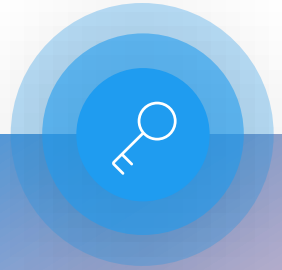
Certificates

Azure Key Vault actions

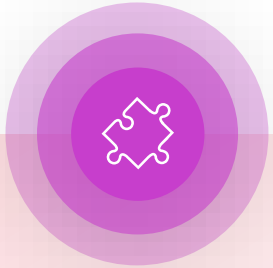


```yaml
steps:
  - uses: actions/checkout@v2

  - uses: Azure/login@v1
    with:
      creds: ${{ secrets.AZURE_CREDENTIALS  }}

  - uses: Azure/get-keyvault-secrets@v1.0
    with:
      keyvault: "TailwindTraders-ADO40-KV"
      secrets: 'DockerPassword'
    id: kvSecretAction

  - uses: Azure/docker-login@v1
    with:
      login-server: tailwindtradersado40.azurecr.io
      username: tailwindtradersado40
      password: ${{ steps.kvSecretAction.outputs.DockerPassword }}
```

Future ideas for Tailwind Traders



Rotate credentials in Key Vault when an admin asks a Teams Bot



Automatically build a new dev environment and fork a repository based on a properly-formatted issue

Pull commit messages since last production release and build release notes for customers

Summary

Add CD to your pipeline!

Continuously deploy somewhere

Protect production with GitHub Environments

Centralize Secrets Storage

Microsoft

Incorporate Performance testing in your CI/CD pipeline

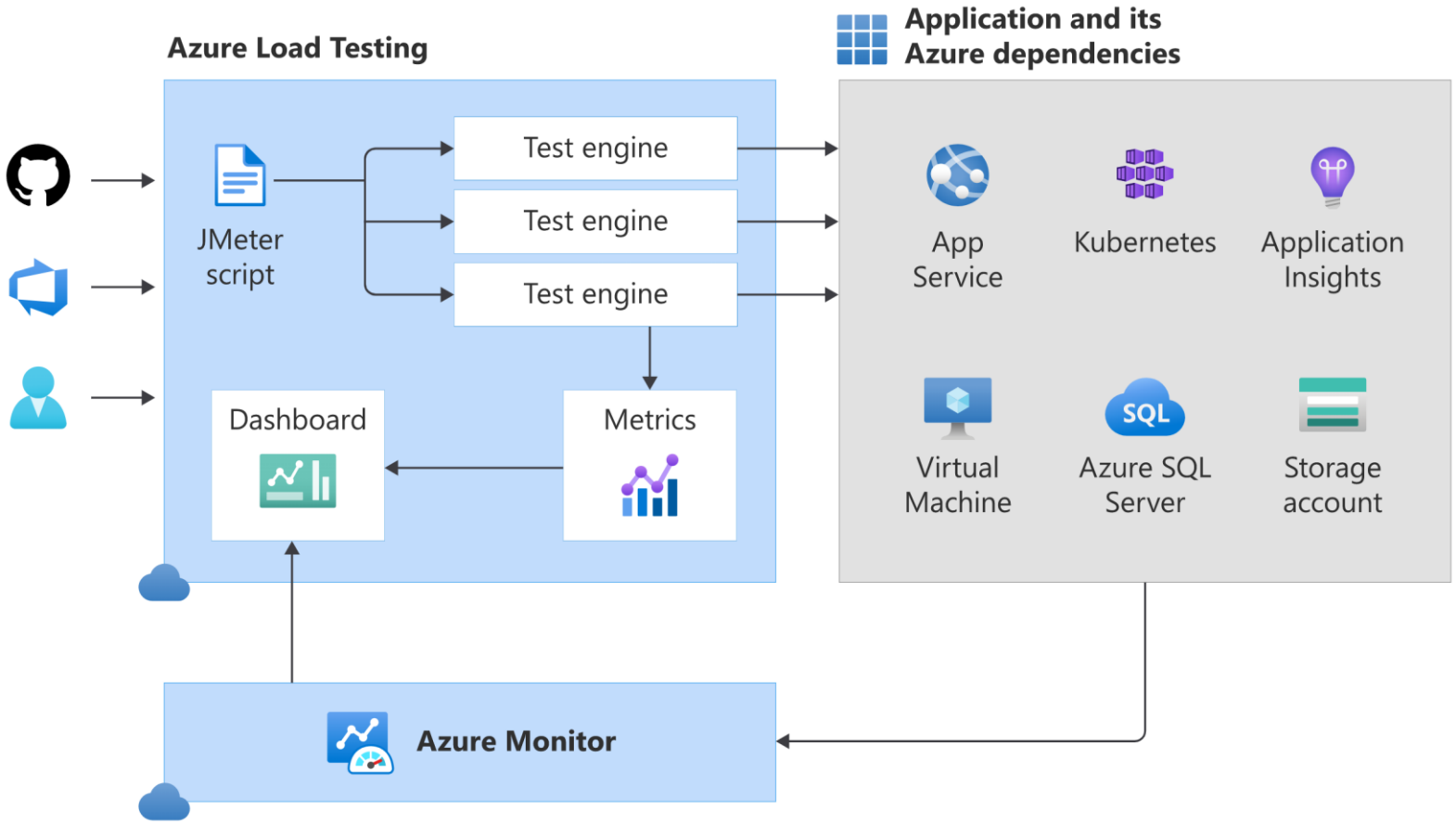
DevOps Learning Path

- Getting Started with DevOps
- Managing the Flow of Work
- Shift security “left” in your CI/CD process
- Delivering changes to cloud
- **Performance Monitoring and maintenance**

Explore Azure Load Testing

Azure Load Testing Preview is a fully managed load-testing service that enables you to generate high-scale load.

Integrate Azure Load Testing in your CI/CD pipeline during the development lifecycle.



Demo

Identify performance regressions with Azure Load Testing Preview and GitHub Actions

Agenda

Understanding Application Behavior

Creating Visibility in Production

Responsible Incident Response

If no one is on-call, everyone is

Guiding Incident Response with Automation

Using computers to help people do the right things

Understanding application behavior in production

The goals for monitoring our production systems

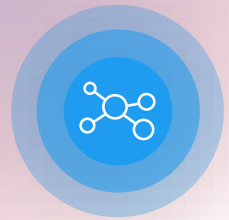
Improve Time to Detect

Reduce Time to Mitigate/Remediate

Enable Validated Learning

Sources of Monitoring Data

Real User Monitoring



Synthetic Transactions

Telemetry

Defining effective monitoring

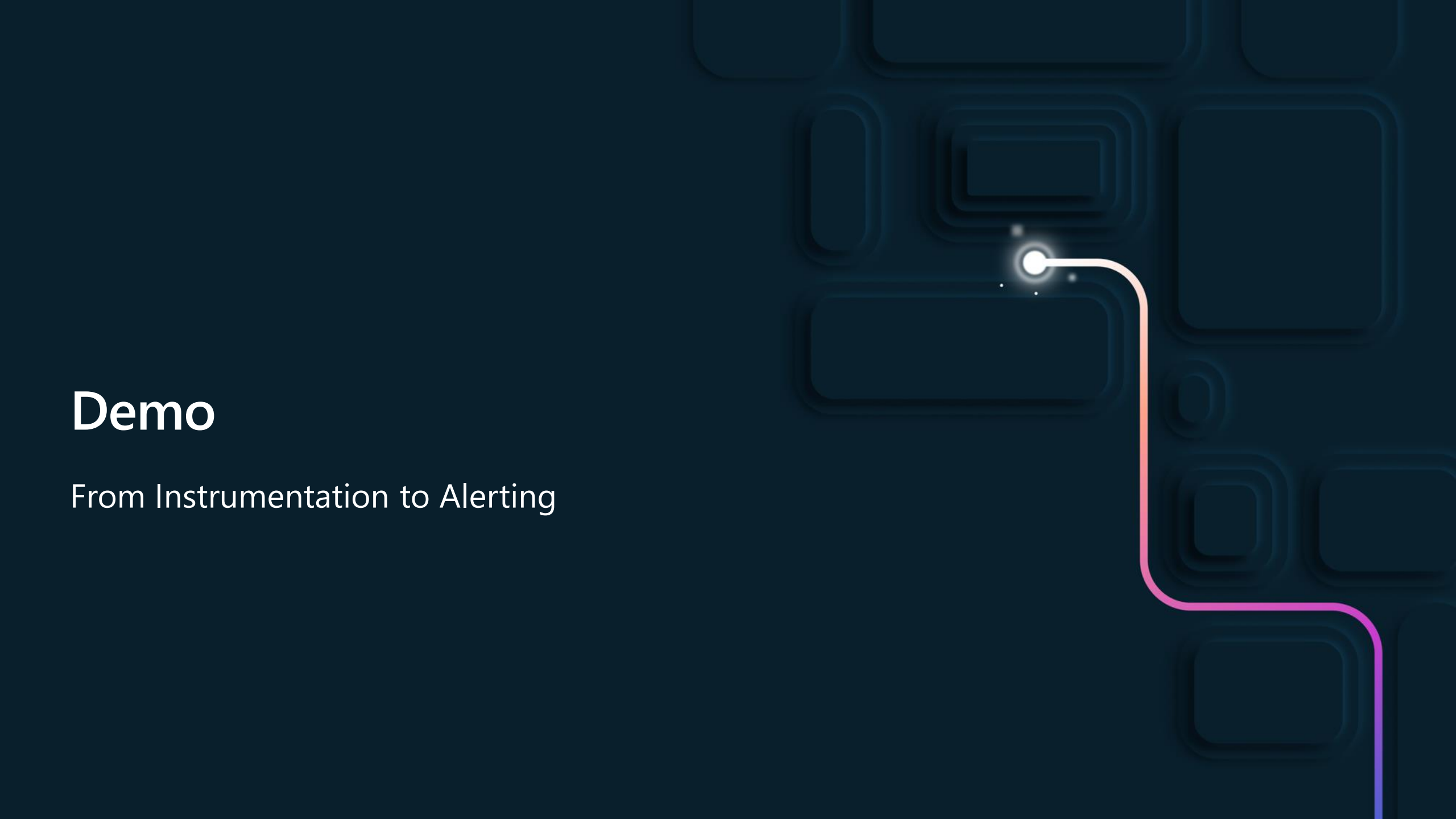
What and why?

Over what time period?

Who needs to know?

Demo

From Instrumentation to Alerting



Responsible incident response

Disaster Strikes!

API calls are failing!

Email notifications
went out… to everyone

And time goes by…

Establishing a basic designated responsible individual rotation

Create a shared DRI schedule

Identify an escalation path

Define response time targets

The DRI is responsible for

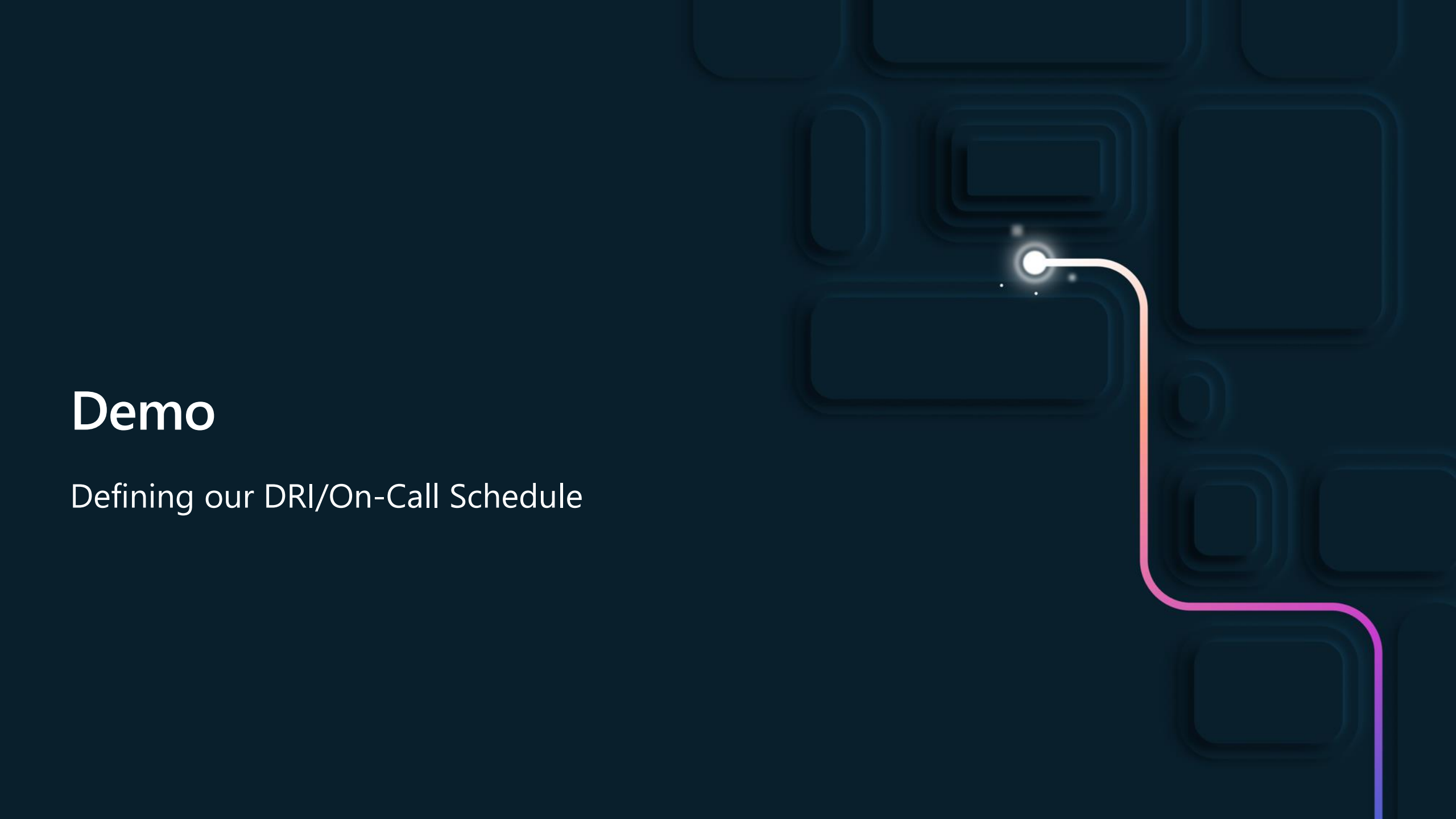
Responding to alerts/incidents in the defined time window

Coordinating with partner service DRIs

Ensuring the proper escalation of severe or long-running issues

Demo

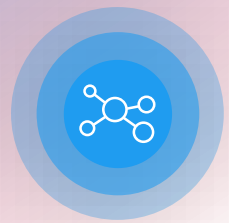
Defining our DRI/On-Call Schedule



Incident Response

Incident Notifications are not:

Broadly distributed



Informational only

Heartbeat or logging

Incident Response

Incident Notifications are not:

Specifically directed

Actionable

In need of human intervention

Guiding incident response with automation

The value of automation

Can't see the value of automation?!! At some point it's an IQ test.

– Jeffrey Snover
Technical Fellow and Creator of PowerShell
2010

Reducing manual intervention

Move manual, repetitive tasks to automation

Reduce human interaction to reduce variability in response

Protect production by ensuring notification and response

Demo

Creating a Basic DRI Notification

Microsoft Defender for Cloud

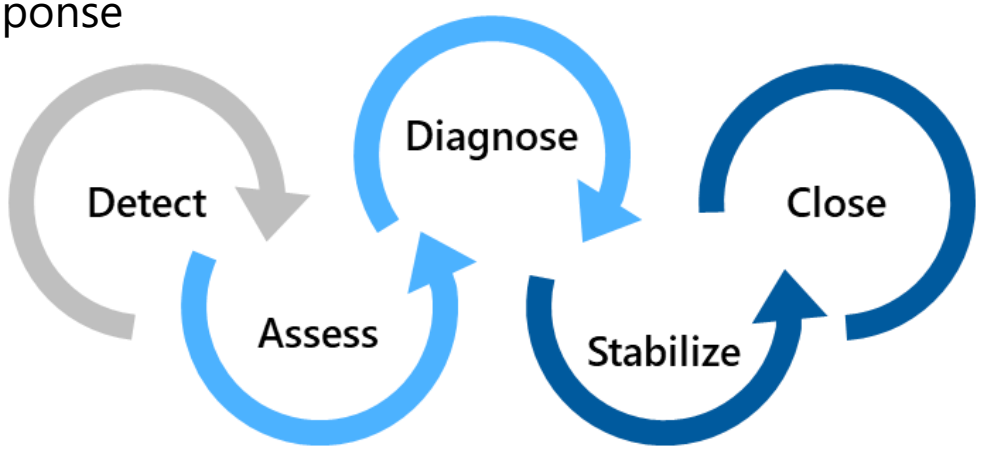
Explore Microsoft Defender for Cloud

Provide security recommendations	Monitor security settings
Analyze and identify potential attacks	Use Azure Machine Learning
Supports Windows and Linux operating systems	Provide just-in-time (JIT) access control

Examine Microsoft Defender for Cloud usage scenarios

Scenario 1

Use Microsoft Defender for Cloud for an incident response



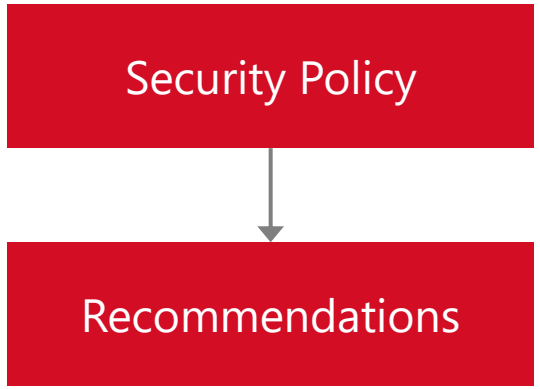
Detect – Verify a high security alert was raised

Access – Obtain information about the alert

Diagnose – Follow the remediation steps

Scenario 2

Use Microsoft Defender for Cloud recommendations to enhance security



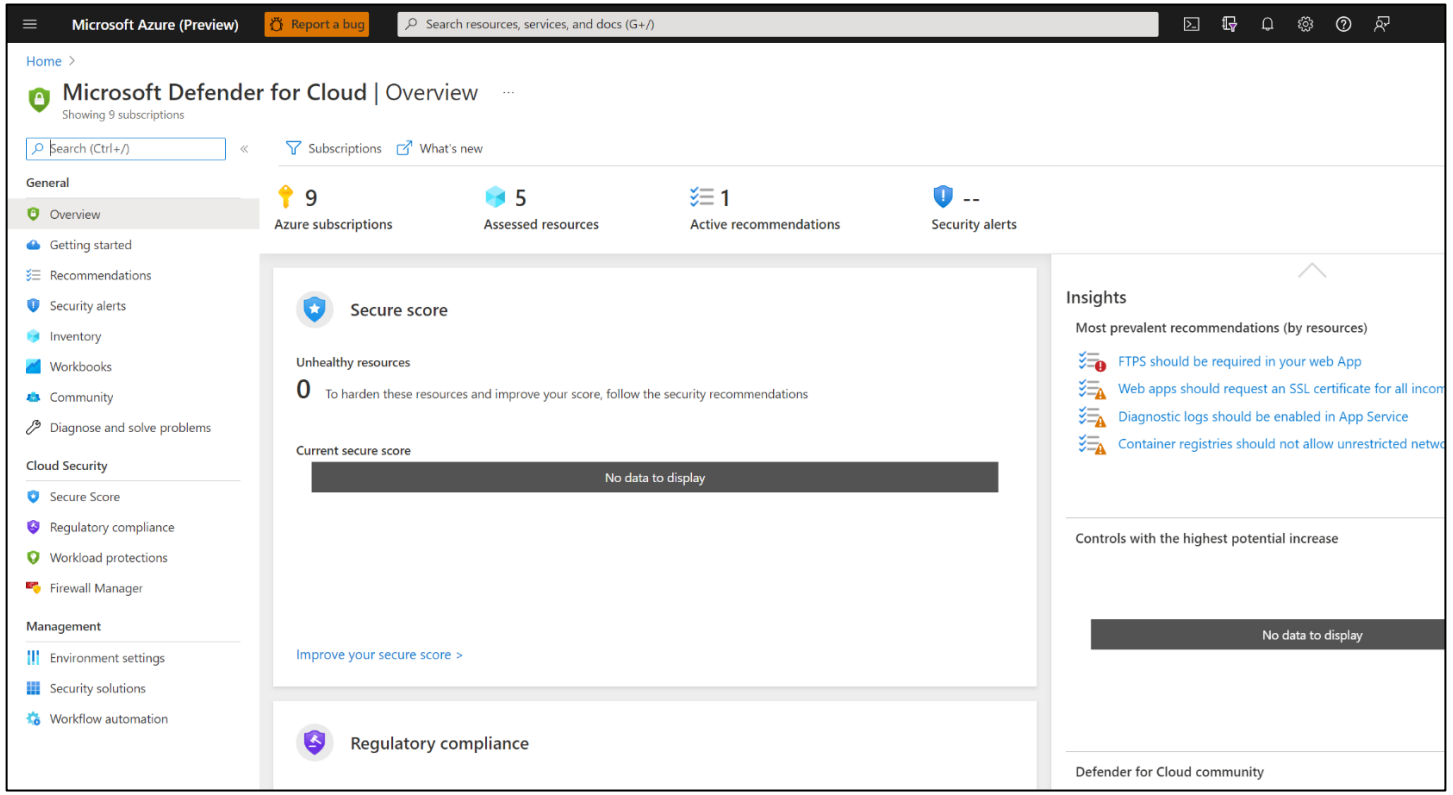
Configure a security policy

Implement the recommendations for the security policy

Understand Microsoft Defender for identity

Identifies, detects, and helps you investigate advanced threats, compromised identities, and malicious insider actions

- Microsoft Defender portal monitors and responds to suspicious activity
- Microsoft Defender sensor monitors domain controller traffic
- Microsoft Defender cloud service connects to Microsoft Intelligent Security Graph



The value of automation

In a lousy economy, NOTHING is more important than automation.

– Jeffrey Snover
Technical Fellow and Creator of PowerShell
2010

Session Resources

Explore Microsoft Learn Content for the AZ-400 Certification

http://aka.ms/AZ-400

Get Certified

Designing and Implementing Microsoft DevOps Solutions

http://aka.ms/AZ-400

Microsoft

Thank you