

Deploying a .NET 10 App to Azure using Bicep and GitHub Actions

What you need

- Required:
 - An editor of some sort
 - GitHub account
 - Some basic Git experience (clone, commit, push, pull)
 - Fork this repo: <https://github.com/scottsauber/workshop-dotnet-azure-github-bicep>
- Email scott@saubersoftware.com if you're doing hands on to get free Azure access
 - Email you will use for Azure
 - GitHub username
- Optional:
 - .NET 10 (<https://dot.net>)
 - Editor/IDE that supports .NET 10 (Rider 2025.3+, VS 2026+, VS Code)
 - Bicep extensions are recommended

What we all need

- Azure to not go down
- GitHub to not go down
- GitHub Actions to not go down
- The conference interest to not go down
- 🙏🙏🙏🙏🙏
- (I do have recordings if needed but that's less fun)

Audience

- Anyone interested in Azure, GitHub, or Bicep
- .NET Developers
- People interested in DevOps, but never got to do it
- People willing to ask questions – ask away – don't wait!

Poll

- How many .NET Developers in the room?
- How many using Azure today?
- How many aren't using Azure?
- How many have used Bicep?
- What about another IAC tool?
- How many are using GitHub Actions today?
- How many are using something else?
- Why are you here? What do you want to learn?

Agenda

- What is the final state of what we're building?
- What is Azure?
- What is Azure App Service? Plans?
- What is Bicep?
- What are GitHub Actions?
- Hands on all throughout
- This is a 2 day workshop, I'm gonna cram into 4 hours
- There is walkthroughs if you fall behind

Goals

- Learn GitHub Actions, Bicep, and Azure
- How they all integrate with a .NET app
- We likely won't get to everything in a few hours
 - This is going to be... a lot
- The feedback loop on this is slow
- Take home a few things back to work, whether beginner or expert

Who am I?

- Director of Engineering at [Lean TECHniques](#)
- [Microsoft MVP](#)
- [Dometrain Author](#)
- Redgate Community Ambassador
- Co-organizer of [Iowa .NET User Group](#)



What are we building?

- .NET 10 API
- Running on Azure App Service
- Configured using Infrastructure as Code with Bicep
- Deployed via GitHub Actions

Features of What We're Building

- Zero Downtime Deployments
- Infrastructure managed by code, not clicking in the Azure Portal
- Automated Build and Deploys
- [Follows Azure Naming Standards for naming resources](#)
- Versioning your app so you know what's deployed*
- Health Checks*
- Secrets in Key Vault, not Source Control*
- Observability using Application Insights*

* Might not get to these in 4 hours.

Why is this important?

- <https://dora.dev/>
- DORA
- We will touch on [6 of the 18 capabilities](#) listed for high performing teams:
 1. Continuous Delivery
 2. Deployment Automation
 3. Flexible Infrastructure
 4. Continuous Integration
 5. Version Control
 6. Monitoring/Observability




Scott Sauber

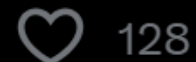
@scottsauber

Setting up a CI/CD pipeline for the first time be like



8:55 PM · Jan 25, 2022

 View post engagements





GitHub Action Runner

- CI (artifacts)

Passes



- Deploy to Dev

Passes



- Deploy to Prod

Deploy



Dev Environment



App Service



Prod Environment



App Service

Deploy

Azure



What is Azure?

- Microsoft offering for cloud hosting
- Offers many services from hosting web apps to databases to caching to messaging to...
- You should probably be picking PaaS offerings (i.e. not VMs)

On-site	IaaS	PaaS	SaaS
Applications	Applications	Applications	Applications
Data	Data	Data	Data
Runtime	Runtime	Runtime	Runtime
Middleware	Middleware	Middleware	Middleware
O/S	O/S	O/S	O/S
Virtualization	Virtualization	Virtualization	Virtualization
Servers	Servers	Servers	Servers
Storage	Storage	Storage	Storage
Networking	Networking	Networking	Networking

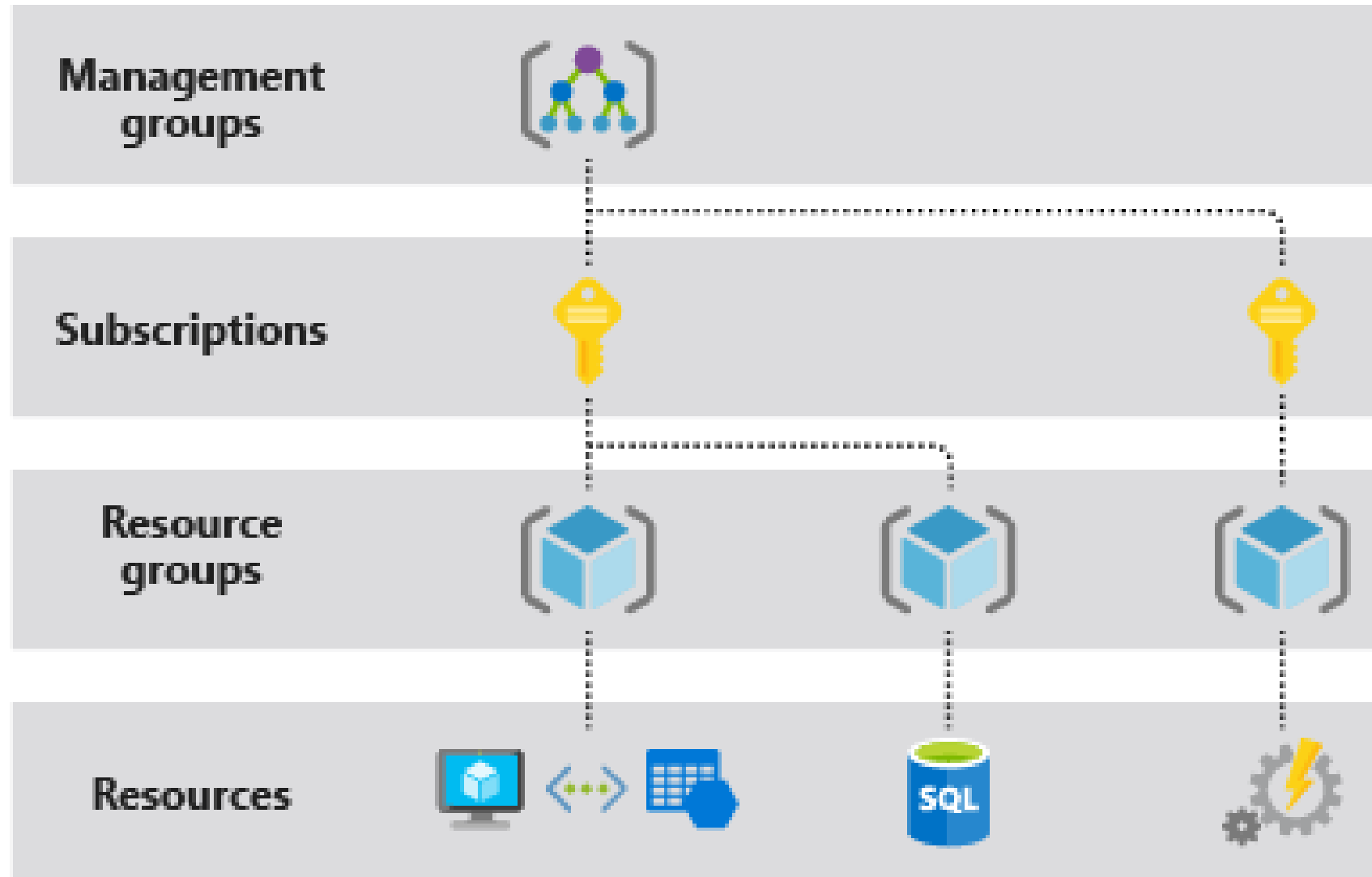
-  You manage
-  Service provider manages

Subscriptions

- Top-ish level organization (ignoring Tenants, Management Groups for a minute)
- Recommended per team per environment
- My default naming convention: sub-<team/dept>-<environment>
 - Eg: sub-accounting-dev
- Role access separation
- Billing separation

Resource Groups

- Related groups of resources (i.e. web, DB, Key Vault, etc)
- Quickly view all resources related to that app
- Conceptually, RG = folder, and Resources = files
- Recommended per app per environment
- Default naming convention: rg-<product name>-<environment>
 - Eg: rg-fancyapp-dev
- May have many RG's in a single subscription
- Role access separation
- Billing separation











Questions?



Azure App Service



What is Azure App Service?

- PaaS offering for hosting applications
-  Handles OS patches, Framework patches
-  Zero downtime deployments with slots
-  SSL Certs
-  Autoscaling
-  Custom Domains
-  Very simple
-  And More
-  Less control because PaaS

What is an Azure App Service Plan?

- Kinda like VM for your App Service(s)
- Pick how much memory, storage, CPU
- Multiple app service on one ASP (should you?)
- Many apps can get away with P0V3 (\$62/mo for Linux)
- Need to be at least Standard to get Slots

\$ an issue?

- Savings Plan – commit to \$ amount
- Save 25% for 1 yr or 45% for 3 yr commitment
- Reservation – commit to compute tier
- Save 35% for 1 yr or 55% for 3 yr commitment



Visits
<https://app-workshop-dnazghbicep-scottsauer-dev.azurewebsites.net>

Dev Environment



App Service



App Service Slot

<https://app-workshop-dnazghbicep-scottsauer-dev-slot.azurewebsites.net>



Visits
<https://app-workshop-dnazghbicep-scottsauer-dev.azurewebsites.net>

Dev Environment



App Service



App Service Slot

<https://app-workshop-dnazghbicep-scottsauer-dev-slot.azurewebsites.net>



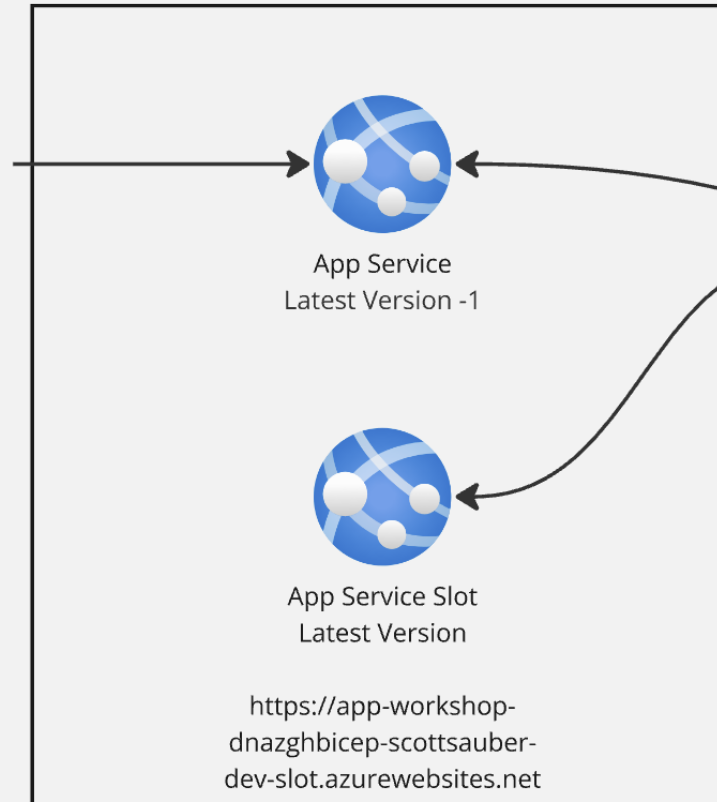
GitHub Action Runner

- Deploy to Slot with latest
- Real App Service has latest -1 version
- Swap the Slot
- Real App Service has latest
- Slot now has latest -1 version



Visits
<https://app-workshop-dnazghbicep-scottsauer-dev.azurewebsites.net>

Dev Environment



GitHub Action Runner

- Deploy to Slot with latest
- Real App Service has latest -1 version
- Swap the Slot
- Real App Service has latest
- Slot now has latest -1 version



Visits
<https://app-workshop-dnazghbicep-scottsauer-dev.azurewebsites.net>

Dev Environment



App Service Slot
Latest Version -1

<https://app-workshop-dnazghbicep-scottsauer-dev-slot.azurewebsites.net>



App Service
Latest Version



GitHub Action Runner

- Deploy to Slot with latest
- Real App Service has latest -1 version
- Swap the Slot
- Real App Service has latest
- Slot now has latest -1 version

Live Demo



Questions?



Bicep



Infrastructure as Code (IaC)

- Source code defining what resources to provision
- Stored in version control
- Declarative – what resources to create, not how to create them
- Deployed via pipeline
- Need a new env? Few lines of code

Without IAC

- Clickety Clack Configuration™ (ClickOps)
- Repeat yourself for each environment
- “It worked in Dev/UAT/Staging, not Prod”
- “It works on my machine”

What is Azure Bicep?

- Used to configure Azure resources
- Built and maintained by Microsoft
- Domain-specific language (fancy word for custom)
- Provides intellisense, error checking, “whatif,” and orders the resource creations
- Built on top of Azure Resource Manager (ARM) – don’t use ARM directly
- Can compose multiple files into “modules”
- Can pass data between modules via outputs
- No state file

What does Bicep look like?

```
resource appServicePlan 'Microsoft.Web/serverfarms@2022-09-01' = {  
  name: 'asp-workshop-demo'  
  location: 'centralus'  
  sku: {  
    name: 'S1'  
  }  
  kind: 'linux'  
}
```

`<> appservice.bicep`

```
1  resource appServicePlan 'Microsoft.Web/serverfarms@2022-09-01' = {  
2    name: 'asp-myapp-dev'  
3    kind: 'linux'  
4    location: 'centralus'  
5    sku: {  
6      name: 'S1'  
7    }  
8  }
```

Create Web App - Microsoft Azure

https://portal.azure.com/#create/Microsoft.WebSite

Microsoft Azure Search resources, services, and docs (G+)

Home > App Services >

Create Web App

Name ☒ .azurewebsites.net

Operating System * ☒ Linux ☐ Windows

Region *

Pricing plan **Standard S1** (100 total ACU, 1.75 GB memory, 1 vCPU)

```
resource appServicePlan 'Microsoft.Web/serverfarms@2022-09-01' = {  
  name: 'asp-workshop-demo'  
  location: 'centralus'  
  sku: {  
    name: 'S1'  
  }  
  kind: 'linux'  
}
```

```
resource appService 'Microsoft.Web/sites@2022-09-01' = {  
  name: 'app-workshop-demo'  
  location: 'centralus'  
  properties: {  
    serverFarmId: appServicePlan.id  
    // others  
  }  
}
```

```
param appName string
@allowed(['dev', 'prod'])
param environment string
param location string
```

```
resource appServicePlan 'Microsoft.Web/serverfarms@2022-09-01' = {
  name: 'asp-${appName}-${environment}'
  location: location
  sku: {
    name: 'S1'
  }
  kind: 'linux'
}
```

dev.bicepparam file

```
using '../main.bicep'  
  
param environment = 'dev'
```

But how do I deploy it?

```
az deployment group create
  --name dev-deployment-1
  --template-file infrastructure/main.bicep
  --parameters infrastructure/environments/dev.bicepparam
  --resource-group rg-some-name-here
  --verbose
```


Key Concepts – Quiz time!

- Resources
- Modules
- Parameters
- .bicepparam
- Outputs
- --whatif

Benefits

- No manual work of configuring in the portal (and repeating for each env)
- Eliminate configuration drift
- Traceability of who, did what, and when
- Give Contributor access to the pipeline – not to individuals

Additional Resources

- Documentation for various Bicep resources:
 - <https://learn.microsoft.com/en-us/azure/templates/microsoft.web/sites?pivots=deployment-language-bicep>

Live Demo



Questions?



Break then Hands On



Hands On for 25 mins

Module 4:

<https://github.com/scottsauber/workshop-dotnet-azure-github-bicep>



CI/CD Pipelines



Continuous Integration

- Automated verification of your application that generates artifacts
- Compiles the app
- Runs the tests
- Independent witness - eliminates “works on my machine”

Continuous Delivery

- Takes the artifacts from CI and deploys them automatically
- Doesn't deploy all the way to Production
- Deploying to Production is a button click

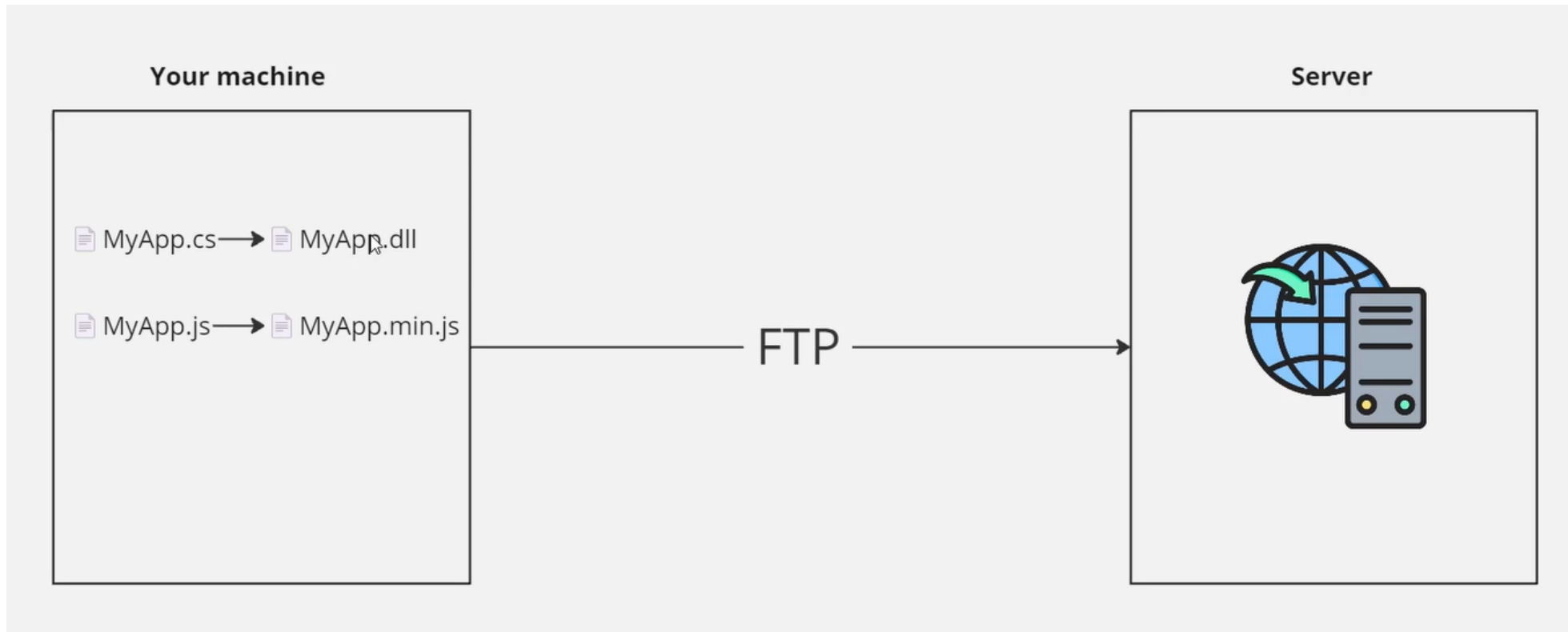
Continuous Deployment

- Deploys all the way to Production automatically
- If the pipeline is green, it's going to Production

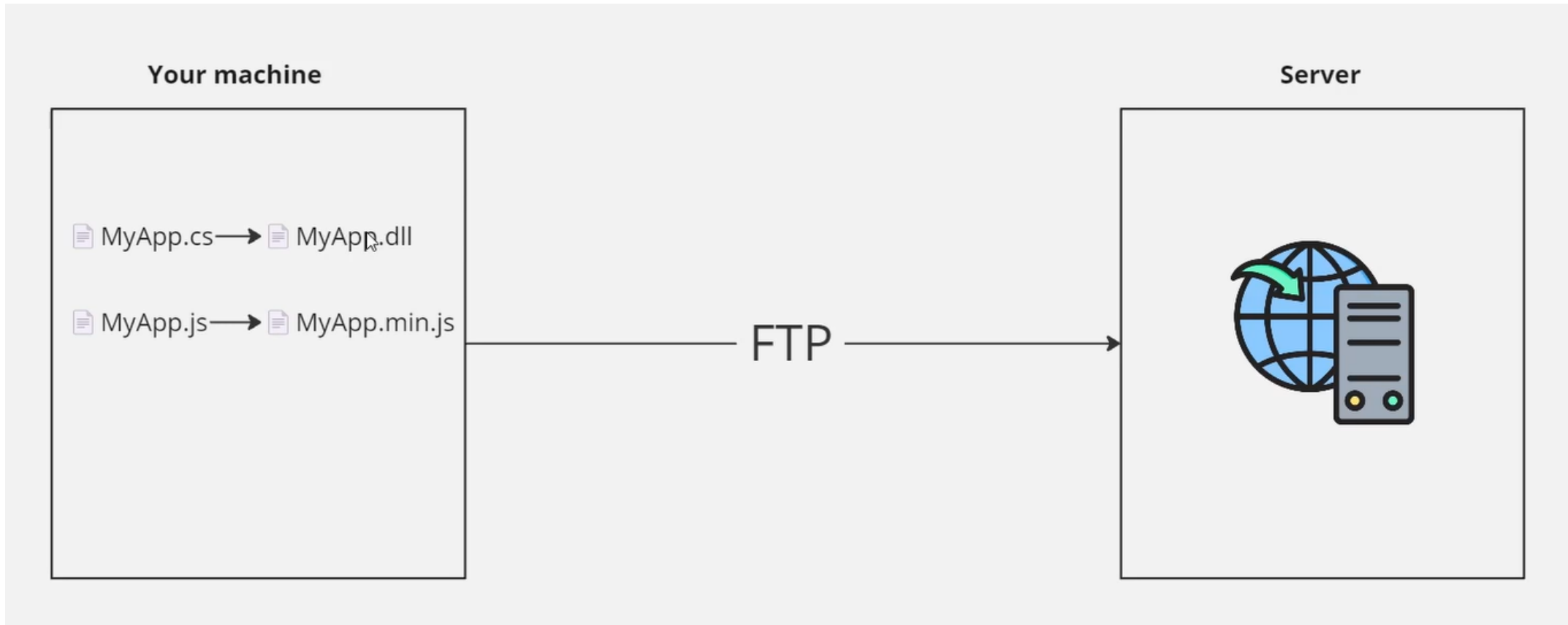
Confident Green

- If our build passes – why aren't we shipping to Production?
- Likely lack of confidence
- Likely missing automated tests or zero downtime deployments, let's fix that
- Ok now why?
- Repeat

Before CI/CD



After CI/CD



What's in a Pipeline?

Continuous Integration

- ✓ Restore Packages
- ✓ Compile
- ✓ Test
- ✓ Format
- ✓ Linting
- ✓ Security Scans
- ✓ Upload Artifacts
- ✓ Alerting on Failure

Continuous Delivery/Deployment

- ✓ Download Artifacts
- ✓ Deploy Artifacts
- ✓ Zero Downtime
- ✓ Deploy IAC
- ✓ Smoke Tests
- ✓ Alerting on Failure

GitHub Actions



What is GitHub Actions?

- Thing doer on a trigger
- Trigger could be PR, push to main branch, open an issue, etc
- Automatically build and deploys your application
- Including the infrastructure (i.e. Bicep) and Database Migrations

Concepts

- Workflows
- Triggers
- Jobs
- Steps
- Inputs
- Secrets

What does GitHub Actions Look Like?

```
1  name: CI - Deploy App and Bicep
2
3  on:
4    push:
5      branches: [main]
6    workflow_dispatch:
7
8  jobs:
9    build_and_test:
10     runs-on: ubuntu-latest
11     name: Build, Test, Upload Artifact
12
13     steps:
14       - name: Checkout repo
15         uses: actions/checkout@v1
16
17       - name: Run dotnet test
18         run: |
19           dotnet test -c Release
20
```

How do I reuse workflows?

```
1  name: CI - Deploy App and Bicep
2
3  on:
4    push:
5      branches: [main]
6    workflow_dispatch:
7
8  jobs:
9    build_and_test:
10     runs-on: ubuntu-latest
11     name: Build, Test, Upload Artifact
12
13     steps:
14       - name: Checkout repo
15         uses: actions/checkout@v1
16
17       - name: Run dotnet test
18         run: |
19           dotnet test -c Release
20
21       - name: Run dotnet publish
22         run: |
23           dotnet publish ./src/WorkshopDemo/WorkshopDemo.csproj -c Release -o ./publish
```

How do I reuse workflows?

```
1  name: Step - Test and Publish
2
3  on:
4    workflow_call:
5      inputs:
6        project_path:
7          required: true
8          type: string
9
10 jobs:
11   build_and_test:
12     runs-on: ubuntu-latest
13     name: Build, Test, Upload Artifact
14
15     steps:
16       - name: Checkout repo
17         uses: actions/checkout@v1
18
19       - name: Run dotnet test
20         run: |
21           dotnet test -c Release
22
23       - name: Run dotnet publish
24         run: |
25           dotnet publish ${{ inputs.project_path }} -c Release -o ./publish
```

How do I consume this reusable workflow?

```
1  name: CI - Test and Publish
2
3  on:
4    push:
5      branches: [main]
6    workflow_dispatch:
7
8  jobs:
9    build_and_test:
10      uses: ../.github/workflows/step-build-and-test.yml
11      with:
12        project_path: ./src/WorkshopDemo/WorkshopDemo.csproj
13
```

How do I consume this from another repo?

```
1  name: CI - Test and Publish
2
3  on:
4    push:
5      branches: [main]
6    workflow_dispatch:
7
8  jobs:
9    build and test:
10     uses: my-org-or-username/repo-name/step-build-and-test.yml
11     with:
12       project_path: ./src/WorkshopDemo/WorkshopDemo.csproj
13
```

Live Demo



Questions?



Hands On for 40 mins

Module 6:

<https://github.com/scottsauber/workshop-dotnet-azure-github-bicep>



Bonus: GitHub Environments

- GitHub natively has the concept of Environments
- These environments can have secrets
- You can also set up Protection Rules on environments
- Let's refactor our code to use Environments!

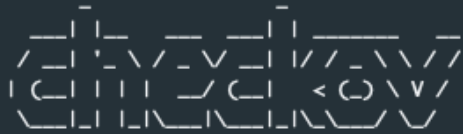
Bonus: PR Checks for Infrastructure

- When you change some Bicep it'd be nice to know what it's going to do
- --whatif for PR's
- Have the --whatif comment back on the PR what it's going to do
- [Example](#)

Bonus: PR Checks for Infrastructure

- Checkov security scanner
- Tells you if something is misconfigured
- ie TLS 1.2 is not the minimum TLS setting for an App Service
- Public Storage Accounts

```
➔ bicep checkov -d /Users/maciejpoborca/Desktop/temp/bicep
[ bicep framework ]: 100%|██████████| [1/1], Current File Scanned=main.bicep
```



By bridgecrew.io | version: 2.1.75
Update available 2.1.75 -> 2.1.87
Run `pip3 install -U checkov` to update

bicep scan results:

Passed checks: 0, Failed checks: 6, Skipped checks: 0

Check: CKV_AZURE_132: "Ensure cosmosdb does not allow privileged escalation by restricting management plane changes"

FAILED for resource: Microsoft.DocumentDB/databaseAccounts.cosmosAccount

Severity: MEDIUM

File: /main.bicep:46-63

Guide: https://docs.bridgecrew.io/docs/bc_azr_storage_4

```
46 | resource cosmosAccount 'Microsoft.DocumentDB/databaseAccounts@2021-04-15' = {
47 |   name: cosmosAccountName
48 |   kind: 'GlobalDocumentDB'
49 |   location: location
50 |   properties: {
51 |     consistencyPolicy: {
52 |       defaultConsistencyLevel: 'Session'
53 |     }
54 |     locations: [
55 |       {
56 |         locationName: location
57 |         failoverPriority: 0
58 |         isZoneRedundant: false
59 |       }
60 |     ]
61 |     databaseAccountOfferType: 'Standard'
62 |   }
63 | }
```

Check: CKV_AZURE_15: "Ensure web app is using the latest version of TLS encryption"

Takeaways

- How to leverage Azure
- Why Infrastructure as Code is useful and how Bicep works
- How to create GitHub Actions and apply IAC to Azure
- Some takeaway tips even if you had experience with this stuff

Resources

- Slides at scottsauber.com (and in the repo)
- <https://github.com/scottsauber/workshop-dotnet-azure-github-bicep>
 - Main has the “final” state of things

Questions?

Contact: ssauber@leantechniques.com

Feedback:



Building and Deploying a .NET 10 App to Azure Using Bicep,
and GitHub Actions

Thanks!

Feedback:



Building and Deploying a .NET 10 App to Azure Using Bicep,
and GitHub Actions