Deploying Azure Functions

Mark Heath
SOFTWARE ARCHITECT

@mark_heath www.markheath.net

Overview

Alternatives to coding in the portal

- Command line tooling
- Visual Studio integration

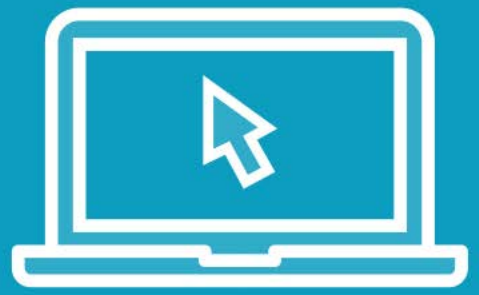
Deployment techniques

- Continuous integration
- Deploying via Git

Three Ways to Create Azure Functions

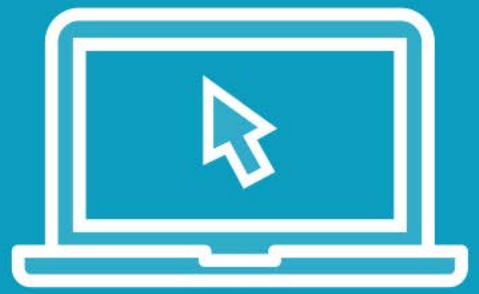
Live	Local	Local
Coding in the Portal	**Command Line Tooling**	**Visual Studio Integration**
Great for prototyping & experiments	npm package	VS2015 extension
No way to rollback	Use your favorite text editor	Currently in preview

Demo



Azure Functions Command Line Tooling

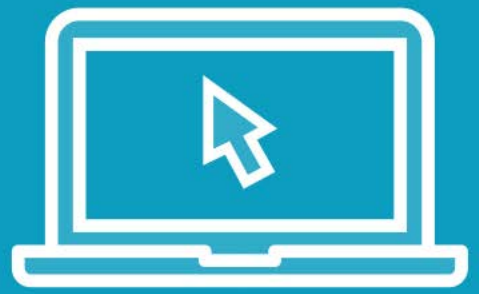
Demo



Using the command line tooling

- Create a new function
- Test it locally

Demo



Using Visual Studio Tooling for Azure Functions

Visual Studio Tooling

Creating

- Function apps
- Functions from templates

Debugging

- Currently C# only

Deploying

- Publish directly from Visual Studio

Azure Functions Deployment Options

Manual Deployment

Kudu, FTP, Web Deploy

Invoke tool manually when you're ready to deploy

Deploy from Visual Studio

Continuous Deployment

Push to Git repository

Code is automatically deployed

Mercurial, DropBox also supported

Git Deployment Options

Local Repository

- Repo hosted for you on Azure

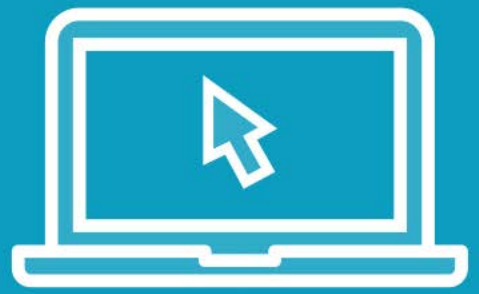
Externally Hosted Repository

- GitHub
- BitBucket
- Visual Studio Team Services

Choose which branch to monitor

- Deploy multiple versions for testing, staging purposes

Demo



Setting up continuous integration

- Local Git repository

Summary

Creating Functions

- Directly in the portal
- Command line tooling
- Visual Studio integration

Deploying Functions

- Manually (e.g. FTP, Visual Studio)
- Continuous integration
- Local or external Git repo
- Ability to roll back

Next Up...

Working in Production