Building Microservices

Mark Heath
CLOUD ARCHITECT

@mark_heath www.markheath.net

Overview

Microservice hosting options

- Benefits of containers

Source control and build

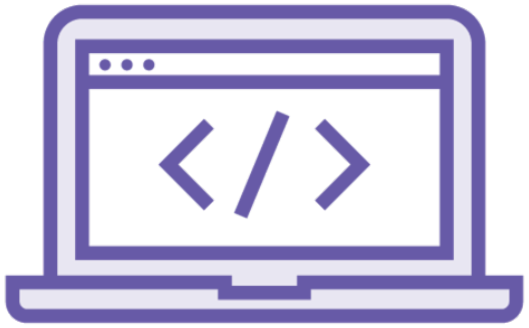
Standard features of a microservice

Service templates

Standard way of working

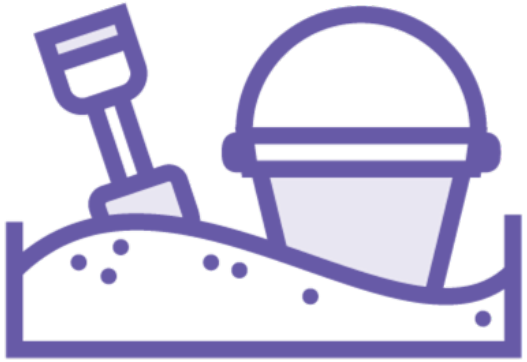
- Developer productivity

Microservice Hosting Environments



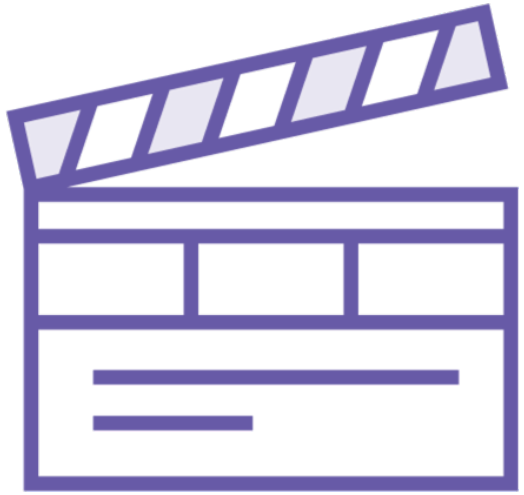
Development

Developers want to debug the application locally



Staging

Testers want to try out the application in a production-like environment



Production

We also need to host the application for end users

Microservices Hosting Options

Virtual Machines

- VM per microservice?
- Operational challenges
- Service discovery

Platform as a Service

- Automatic scale-out
- DNS addresses
- Load balancing
- Security
- Monitoring
- Serverless

Containers

- Portable: run anywhere
- Easily run locally
- Docker Compose

Azure Functions Fundamentals

by Mark Heath

Discover how Azure Functions allows you to easily write serverless code in your language of preference to handle events at scale, with minimal overhead and cost.

Microsoft Azure Developer: Create Serverless Functions

by Mark Heath

Azure Functions is the quickest and easiest way to get your code running in Azure. This course will teach you how to create your own serverless functions, integrate with other services, and host them in Azure or Docker containers.

Building Serverless Applications in Azure

by Mark Heath

Over the years serverless has become a buzzword, but what does it look like to build via a serverless architecture? This course will teach you how to build serverless applications in Azure, from implementing web hosting to deployment and monitoring.

Demo

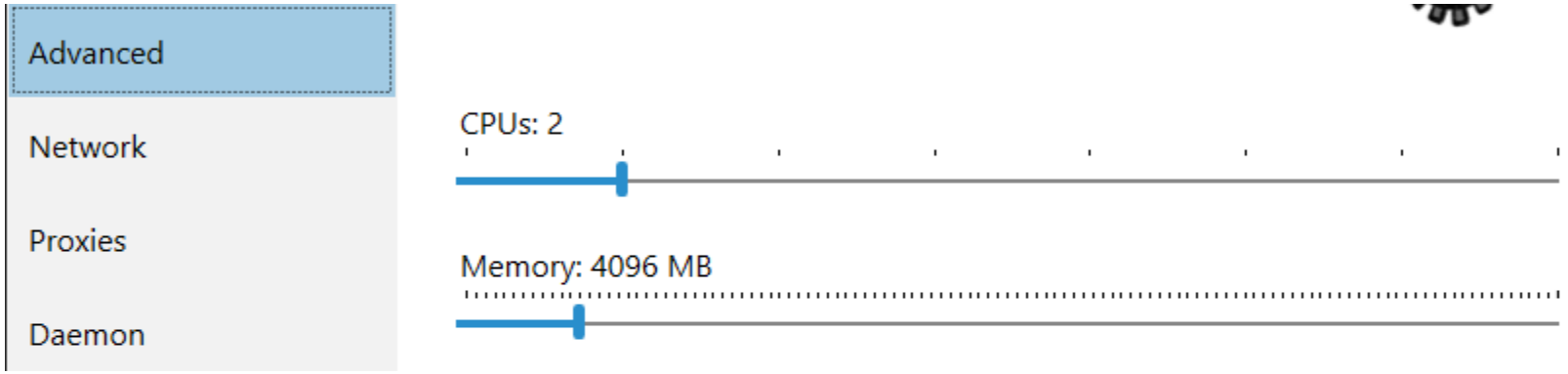
Running eShopOnContainers locally

Development Environment Setup

1. **Install Docker Desktop (Windows or Mac)**
   https://www.docker.com/products/docker-desktop

2. **Clone eShopOnContainers source code**
   git clone https://github.com/dotnet-architecture/eShopOnContainers.git

3. **Increase Docker Desktop available memory to 4GB (in Advanced Settings)**



4. **Configure Windows firewall rules (using supplied PowerShell script)**
   .\cli-windows\add-firewall-rules-for-sts-auth-thru-docker.ps1

eShopOnContainers Instructions

Setting up your development environment for eShopOnContainers

Windows based (CLI and Visual Studio)

https://github.com/dotnet-architecture/eShopOnContainers/wiki/Windows-setup

Mac based (CLI ans Visual Studio for Mac)

https://github.com/dotnet-architecture/eShopOnContainers/wiki/Mac-setup

https://github.com/dotnet-architecture/eShopOnContainers

Why Use Containerized Microservices?

Building microservices individually is time-consuming

- Install dependencies
- Set up configuration

Trivially start everything with Docker Compose

- Can debug code running in containers

Containers are not required

- But automate developer processes

Creating a New Microservice

Source control repository per microservice

- Avoid tight coupling between services

Continuous integration build

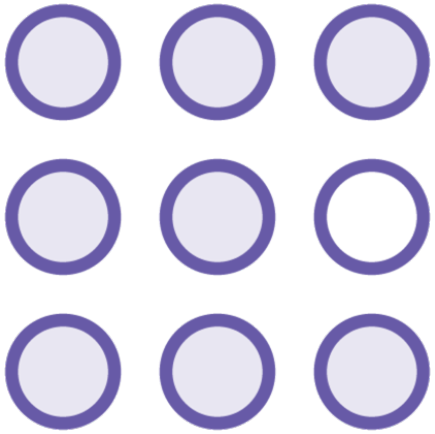
- Run automated tests

Types of Test

Unit Tests

Fast to run

High code coverage



Service-level Tests

Test a single service in isolation

Mock collaborators

End-to-end Tests

Production-like environment

Can be fragile

Consider using a microservice template or “exemplar”

Standardizing Microservices

Logging

Health checks

Configuration

Authentication

Build scripts

Benefits of Service Templates

Reduced time to create a new microservice

Consistent tooling (but still allow for best tool for the job)

Increased developer productivity

Ability to run the microservice in isolation

Run in context of full application
- Locally (e.g. Docker Compose)
- Connected to shared services

Summary

Hosting microservices

Using containers

Source control and build

Automated tests

- Unit tests
- Service-level tests
- End-to-end tests

Standardizing microservices

- Service template

Up next...

Communicating between microservices