Agile Dev
Better Software
DevOps WEST
A TECHWELL EVENT

DW2

Docker Containers
Wednesday, June 6th, 2018, 11:30 AM

Managing Microservices Using Terraform, Docker, and the Cloud

Presented by:

Derek Ashmore
Asperitas Consulting

Brought to you by:

350 Corporate Way, Suite 400, Orange Park, FL 32073
888-268-8770 ∙ 904-278-0524 - info@techwell.com - https://www.techwell.com/

Derek Ashmore

Asperitas Consulting

Derek Ashmore is a senior technology expert with more than thirty years of experience in a wide variety of technologies and industries. Derek is currently focusing on microservices architectures, cloud computing, and migrating applications to the cloud. Derek's approach allows companies to increase speed to market while also increasing application availability. Derek routinely speaks at JavaOne, DevNexus, the Chicago Coders Conference, and many others. His books include The Java EE Architect's Handbook and Microservices for Java EE Architects.

Managing Microservices

using Terraform, Docker, and the Cloud

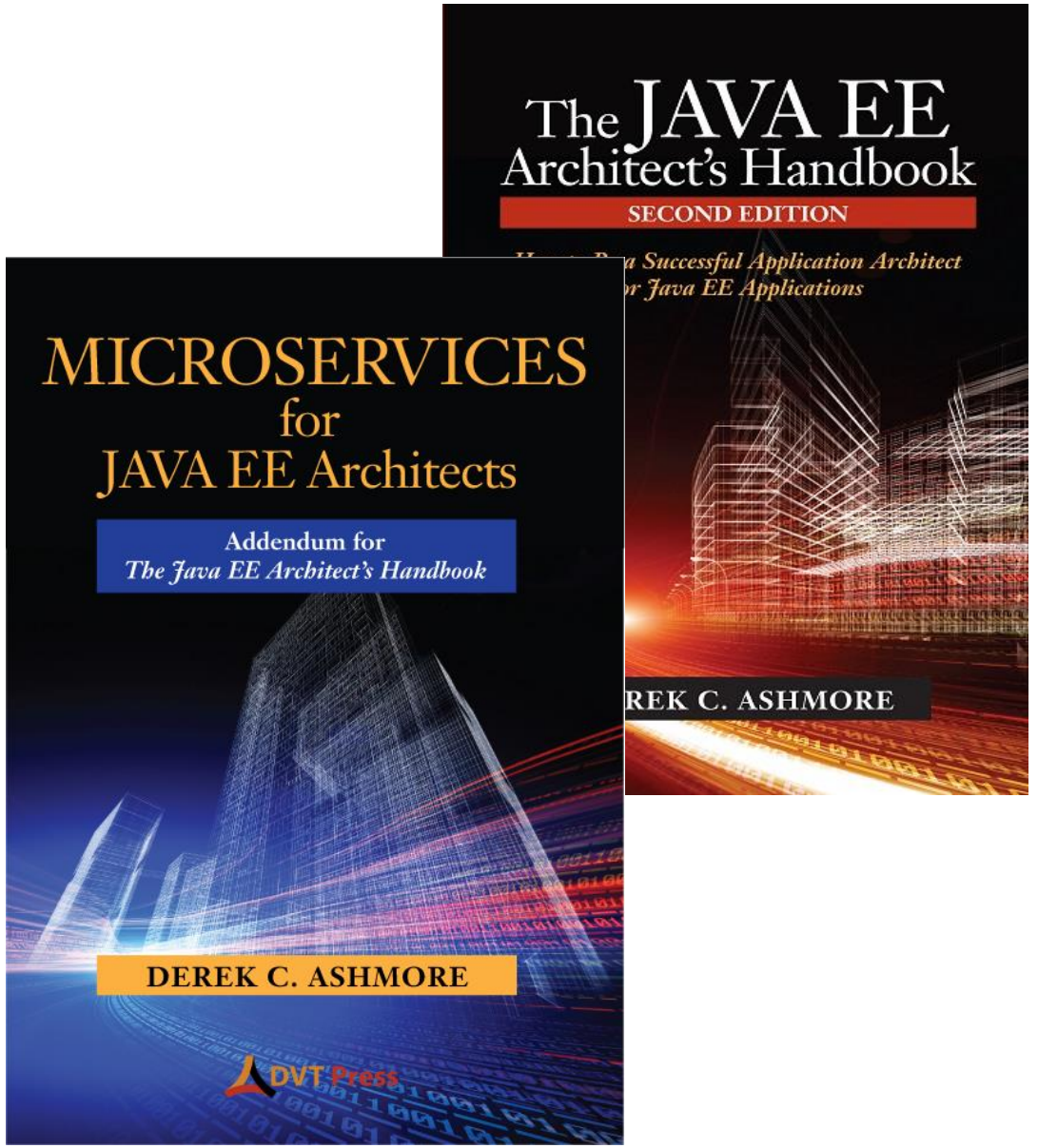
Given by Derek C. Ashmore

DevOps West – June 6, 2018

©2018 Derek C. Ashmore, All Rights Reserved

Who am I?

- Professional Geek since 1987
- Java/J2EE/Java EE since 1999
- AWS since 2010
- Specialties
  - Refactoring
  - Performance Tuning
- Yes – I still code!



ASPERITAS CONSULTING

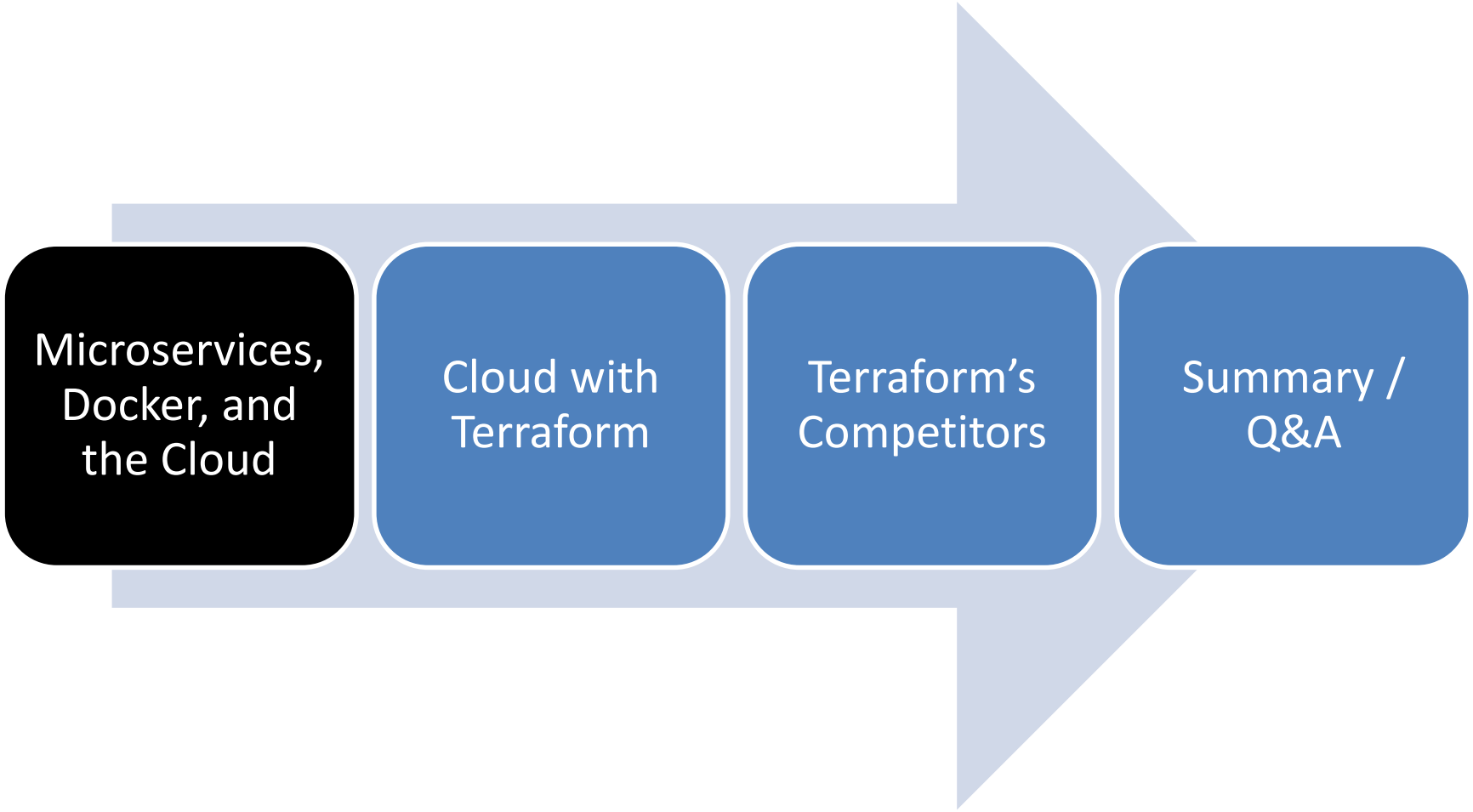
©2018 Derek C. Ashmore, All Rights Reserved

Discussion Resources

- This slide deck
  - http://www.slideshare.net/derekashmore
- The hands-on-lab code
  - https://github.com/Derek-Ashmore/terraform-hands-on-lab
- The Moneta microservice (written in Java)
  - https://github.com/Derek-Ashmore/moneta
- Slide deck has hyper-links!
  - Don’t bother writing down URLs

©2018 Derek C. Ashmore, All Rights Reserved

Agenda



©2018 Derek C. Ashmore, All Rights Reserved

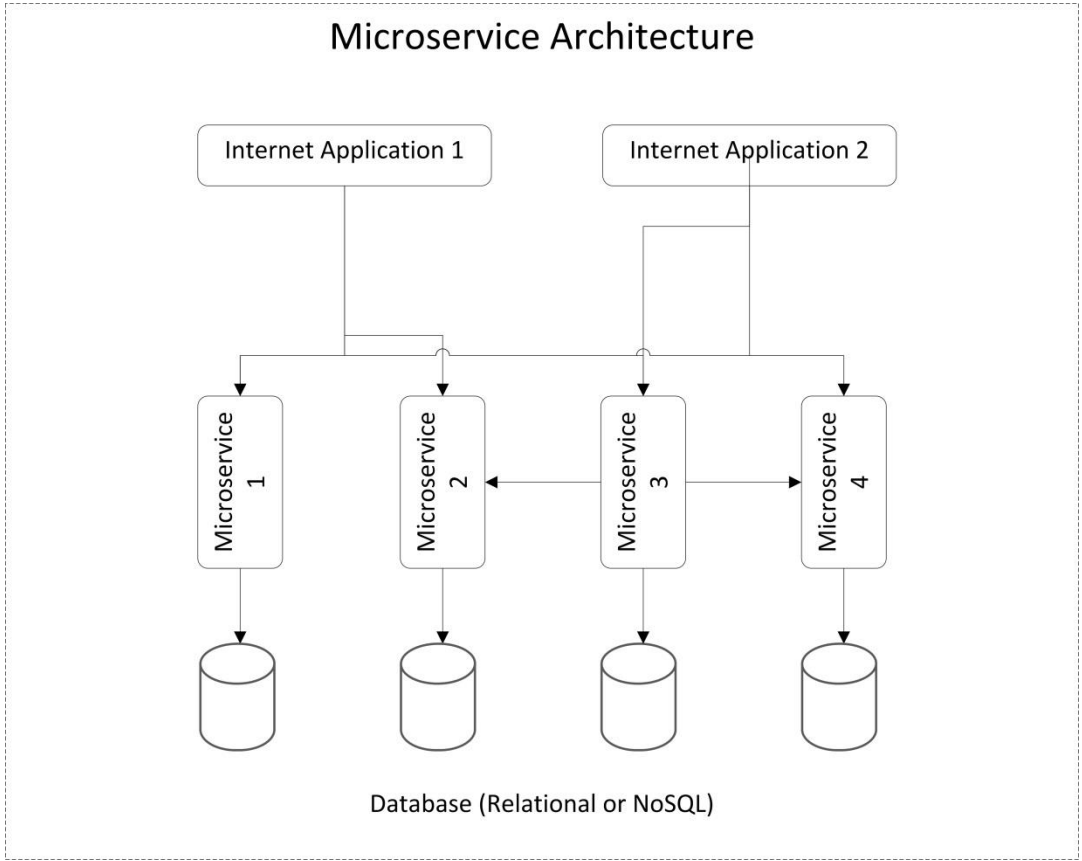
What are Microservices?

- No concrete definition
- Common microservice traits
  - Single functional purpose
    - Most/all changes only impact one service
    - Not dependent on execution context
      - “loosely coupled”
  - Independent process/jvm
  - Stateless
  - Standard Interface (typically Web Service/REST)
  - Analogy: Stereo system, Linux utilities

©2018 Derek C. Ashmore, All Rights Reserved

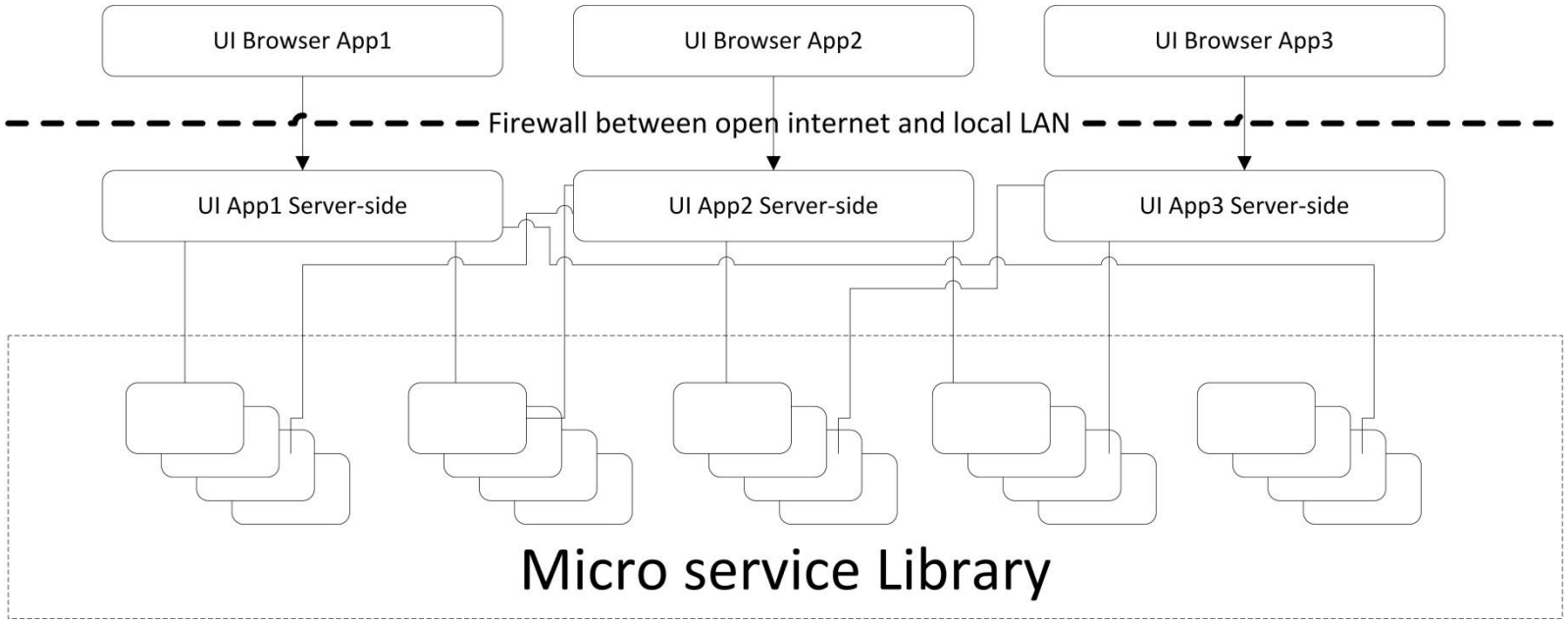
Microservices Application Architecture

- Separate Databases
- Eventual Consistency
- More network activity



©2018 Derek C. Ashmore, All Rights Reserved

Typical Microservice Library

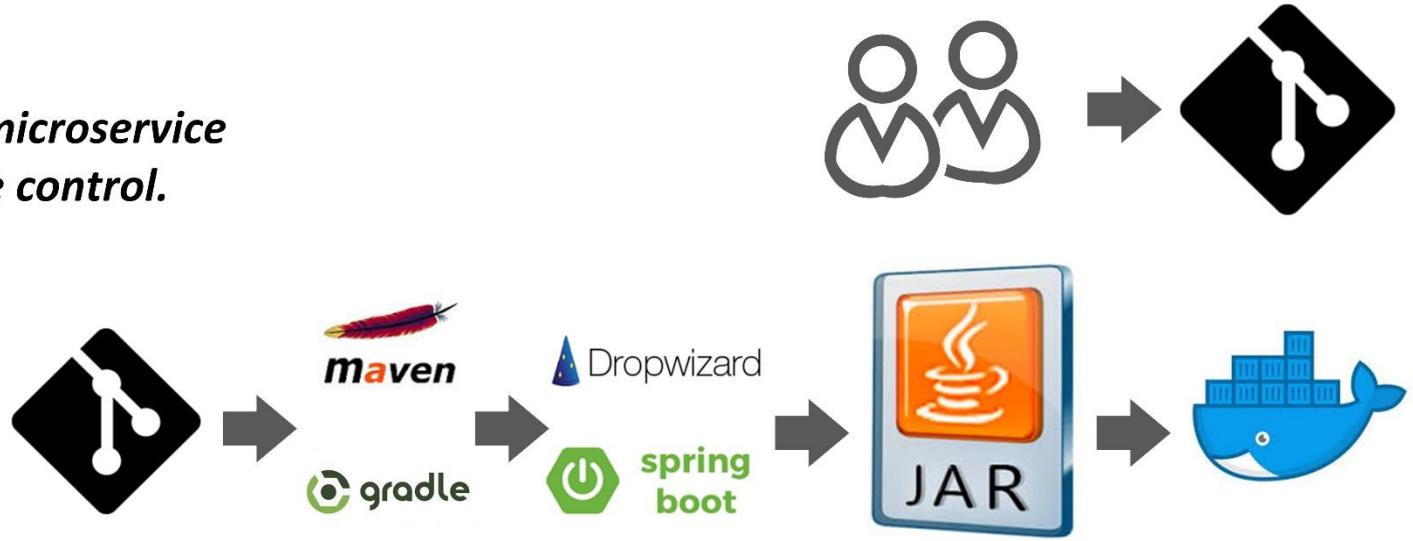


©2018 Derek C. Ashmore, All Rights Reserved

Microservice Development

Developers check microservice code into source control.

Build produces Docker image



©2018 Derek C. Ashmore, All Rights Reserved

Docker

- Is a “mini VM”
  - runs a linux kernal
- Compare to shipping container
- Standard “connections” to outside world
- Supported formally by Oracle, Tomcat, Jboss, and many more

Package Once, Run Anywhere!

©2018 Derek C. Ashmore, All Rights Reserved

Why Docker?

- Docker is Win-Win
  - Easier for OPS and system administrators
    - All software looks the same
    - Standard interface for disk and network resources
      - Containers can be “linked”
    - Inherently automated
  - Easier for developers
    - Fewer environment difference issues
    - Less to communicate to OPS / system administrators
    - Easy to leverage work of others (docker-hub)

©2018 Derek C. Ashmore, All Rights Reserved

Microservice Deployments

Build produces Docker image

Docker images are deployed one of several ways

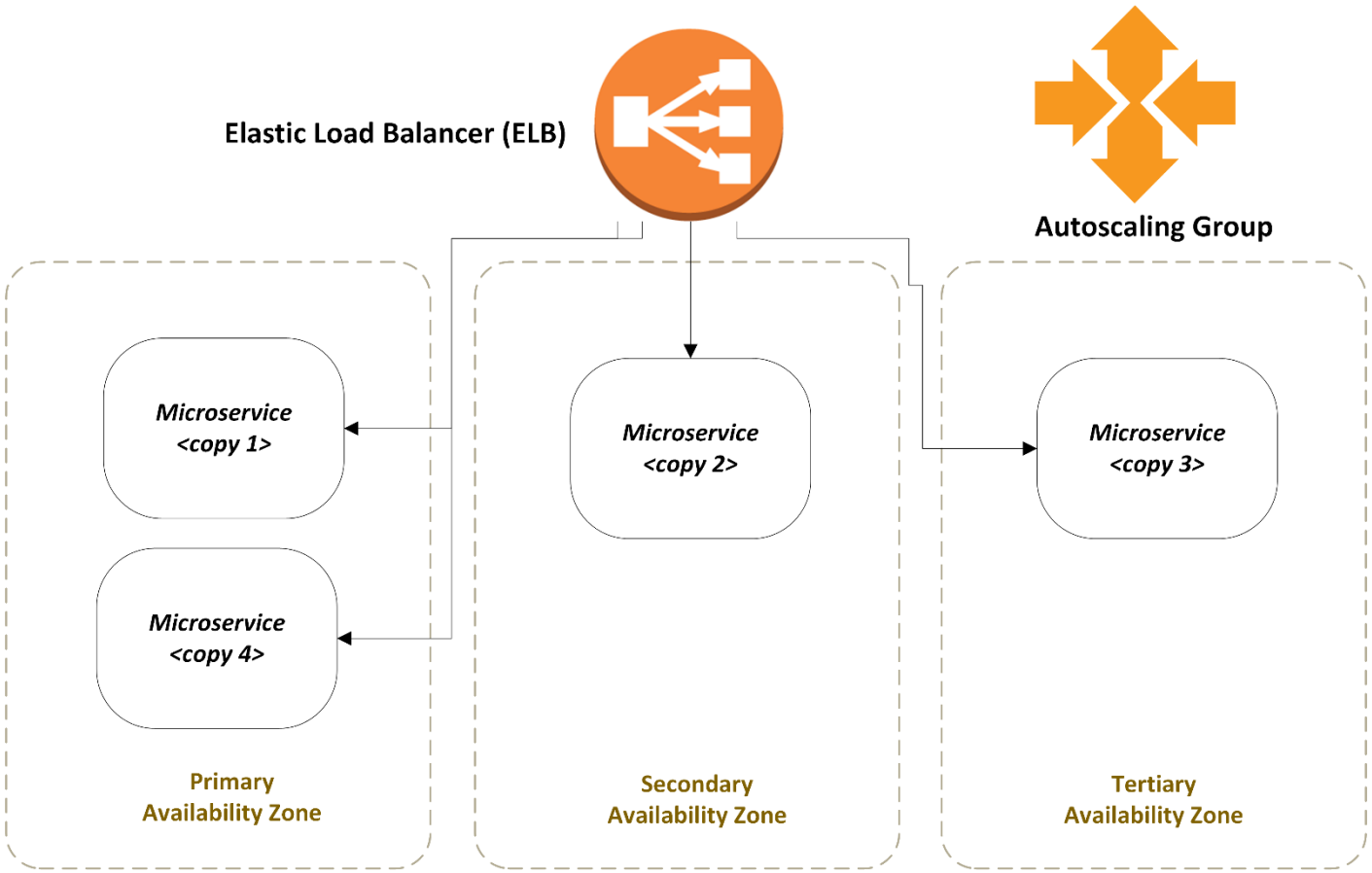
- To the cloud directly
- Through Apache Mesos
- Through Kubernetes
- Through Cloud Foundry

©2018 Derek C. Ashmore, All Rights Reserved

Scalable Deployment at AWS

Basic Microservice Install at AWS

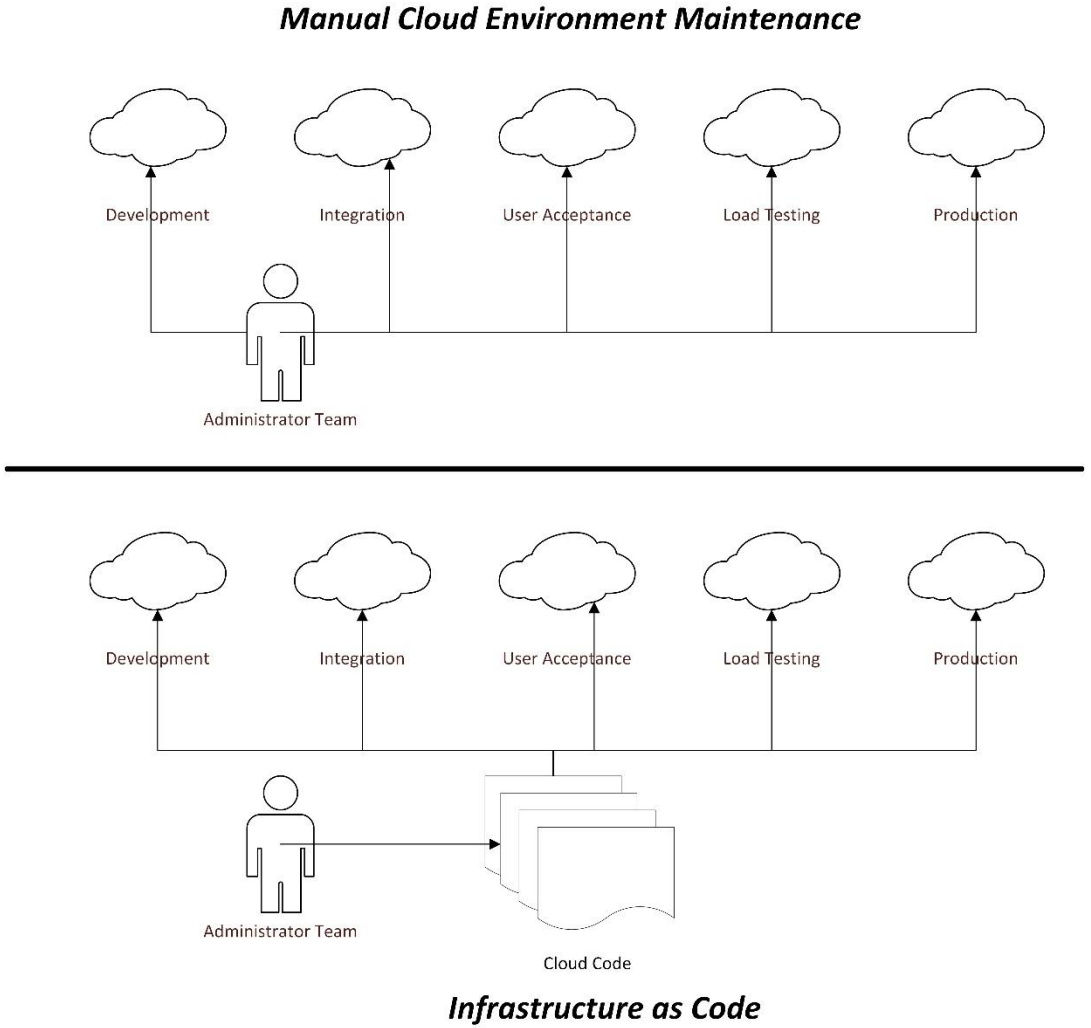
- Horizontal scaling is supported
  - Multiple copies of microservice / web application running at the same time
- Elastic Load Balancer distributes load across copies of your service
  - Sticky sessions available
- ELB can use health checks
- Autoscaling Groups scale number of copies up and down based on rules you give it
  - CPU Utilization or other metrics



©2018 Derek C. Ashmore, All Rights Reserved

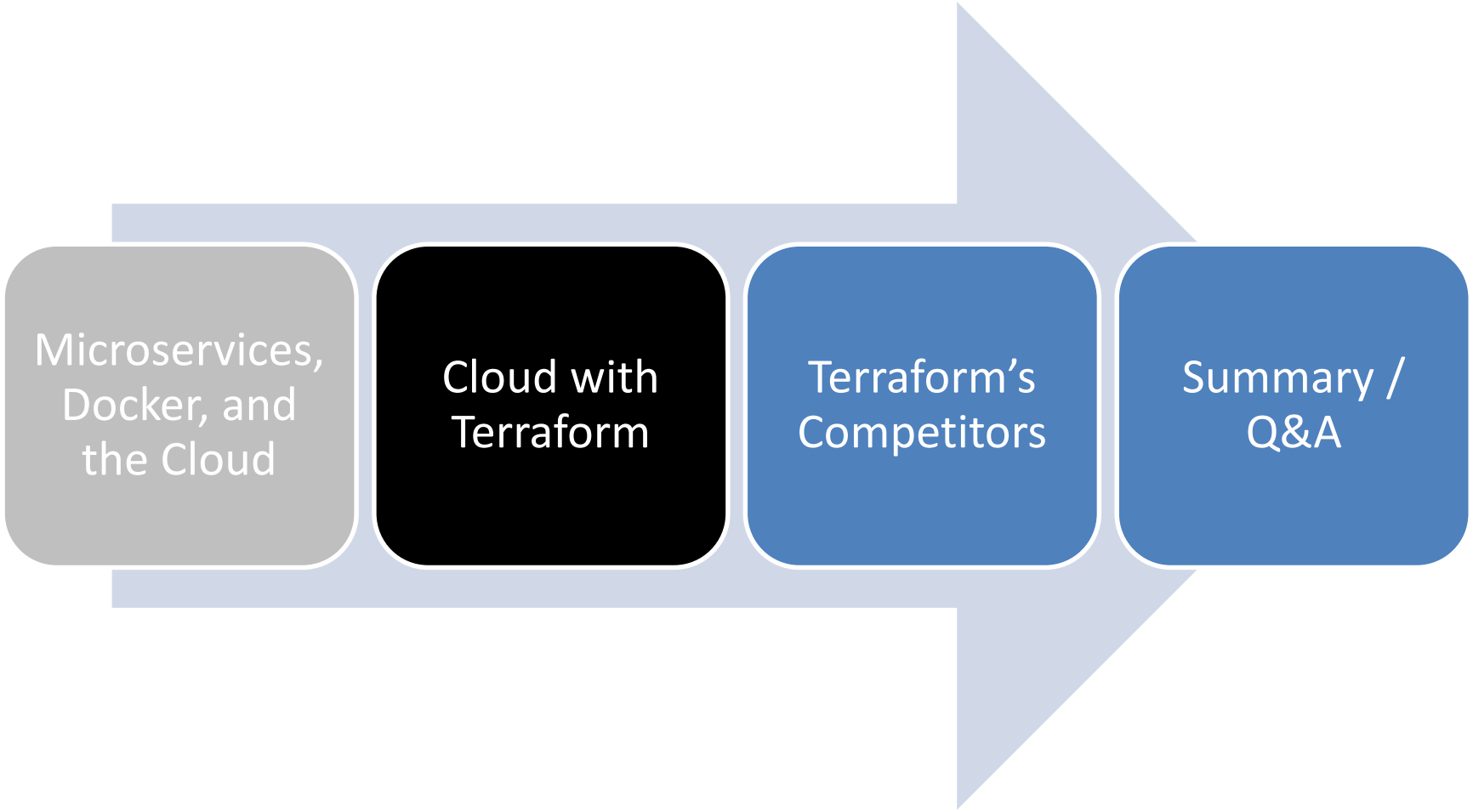
Infrastructure as Code

- Manual changes
  - Increase errors
  - Increase unwanted differences between environments
  - Increase admin workload
- Scripted/Coded changes
  - Larger upfront cost, but…..
  - Less busywork
  - Leverage Others Work
  - Decreases Errors
  - Errors fixed in one place
  - Eliminates unwanted differences
  - Change history (with source control)



©2018 Derek C. Ashmore, All Rights Reserved

Agenda



©2018 Derek C. Ashmore, All Rights Reserved

Terraform

- Cloud Management
  - Open Source
    - **Very** active community
  - Extensible to any cloud vendor
    - AWS, Azure, GCP, AliCloud, Digital Ocean, OpenStack
  - Supported for Cloud Support products
    - Chef, Consul, Kubernetes, Datadog
    - 62 Providers as of April, 2017 and growing

©2018 Derek C. Ashmore, All Rights Reserved

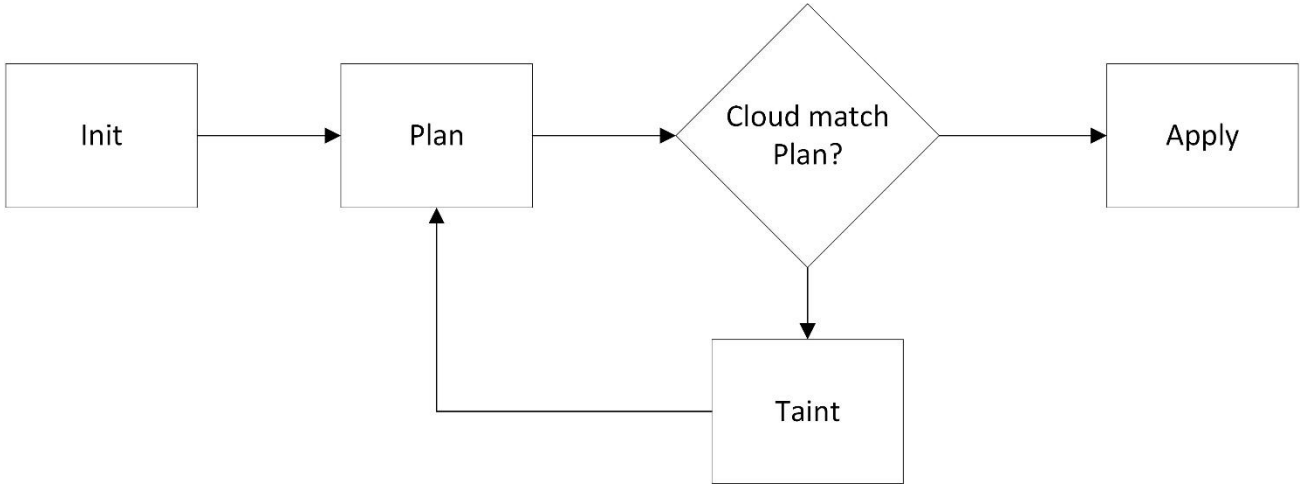
Terraform HCL

- Declarative Language
  - Describe what the end product contains
    - Terraform figures out how to get there
  - Terraform Resources
    - Describes deployed artifacts
      - Network → Virtual Networks, Subnets, Network ACLs, Gateways, ELB/ALB
      - Hosts → Virtual Machines, Databases
      - Security → Security groups/policies/roles/groups/users
      - Much more

©2018 Derek C. Ashmore, All Rights Reserved

Terraform Basics

- Declarative Programming
  - All *.tf files loaded → Terraform decides execution order
  - No GUI → All command line and text editor
- Terraform Command Flow



©2018 Derek C. Ashmore, All Rights Reserved

Terraform Resources

```
# Define three public subnets across different avaialbility zones. Firewalls go here.
# Public traffic allowed in; firewall determines/logs what passes beyond.
resource "aws_subnet" "public" {
  count = 3
  vpc_id = "${aws_vpc.newVPC.id}"
  availability_zone = "${data.aws_availability_zones.available.names[count.index]}"
  cidr_block = "${var.cidr_block_prefix}${var.cidr_block_public_subnet_segment_suffix_list[count.index]}"
  map_public_ip_on_launch = "true"

  tags {
      Name = "${var.vpc_name}.public${count.index}"
      Scope = "Public"
  }
}
```

- AWS Subnet Resource
  - `Count = 3` → Three subnets created
  - Availability Zones come from a data source (lookup)
  - CIDR blocks are input variables
- Sample source

©2018 Derek C. Ashmore, All Rights Reserved

Terraform Data Sources

```
# List of currently defined availability zones in the current region.
data "aws_availability_zones" "available" {}

data "aws_subnet" "dmzSubnet0" {
  vpc_id = "${data.aws_vpc.targetVpc.id}"
  filter {
    name = "tag:Name"
    values = ["${data.aws_vpc.targetVpc.tags.Name}.dmz0"]
  }
}
```

- Example Data Sources (lookups)
- Sample source

©2018 Derek C. Ashmore, All Rights Reserved

Terraform Providers

```
provider "aws" {
  access_key = "${var.aws_key}"
  secret_key = "${var.aws_secret_key}"
  region     = "${var.aws_region}"
}
```

```
provider "azurerm" {
  subscription_id = "..."
  client_id       = "..."
  client_secret   = "..."
  tenant_id       = "..."
}
```

- Example Provider
- Sample AWS source
- Azure Provider

©2018 Derek C. Ashmore, All Rights Reserved

Terraform Input Variables

```
variable "aws_key" {}
variable "aws_secret_key" {}
variable "aws_region" {
        default = "us-east-1"
}
```

- Example Provider
- Sample source

```
variable "cidr_block_private_subnets" {
  type    = "list"
        description = "CIDR blocks of Private subnets"
  default = ["10.0.20.0/24", "10.0.21.0/24", "10.0.22.0/24"]
}
```

©2018 Derek C. Ashmore, All Rights Reserved

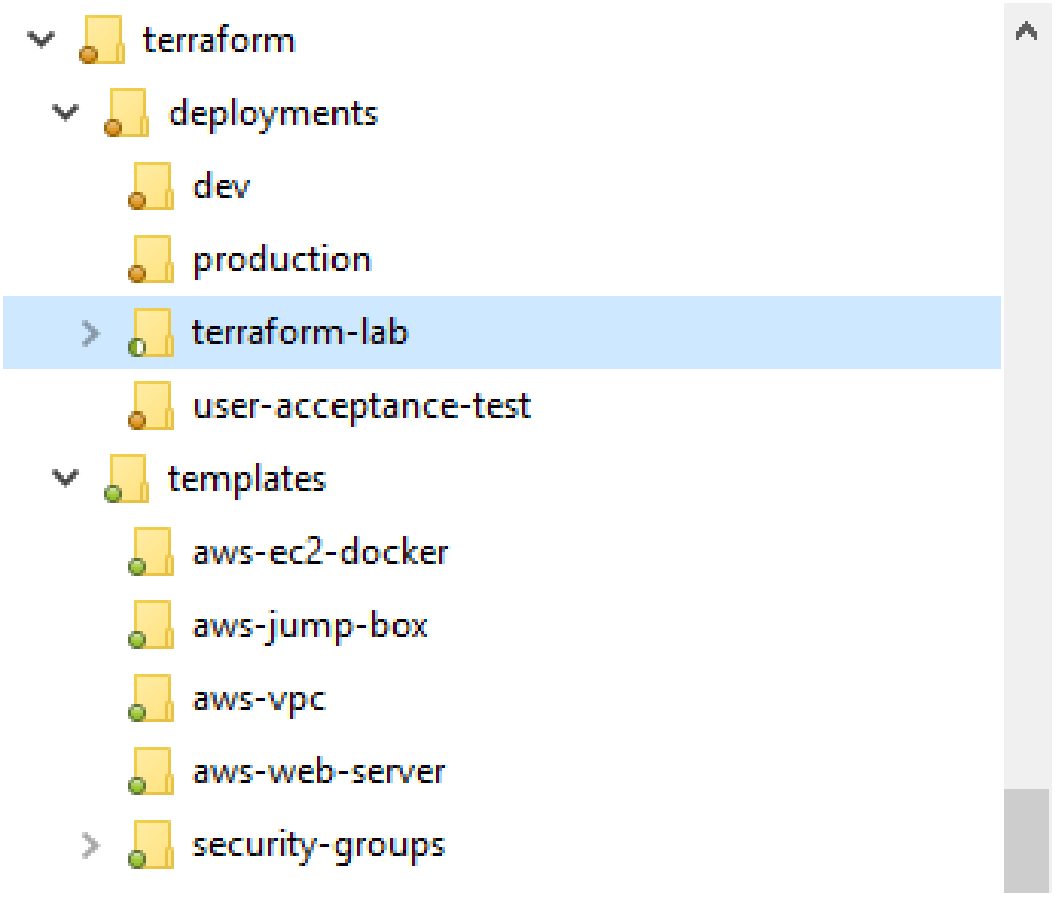
Reusing Terraform Templates

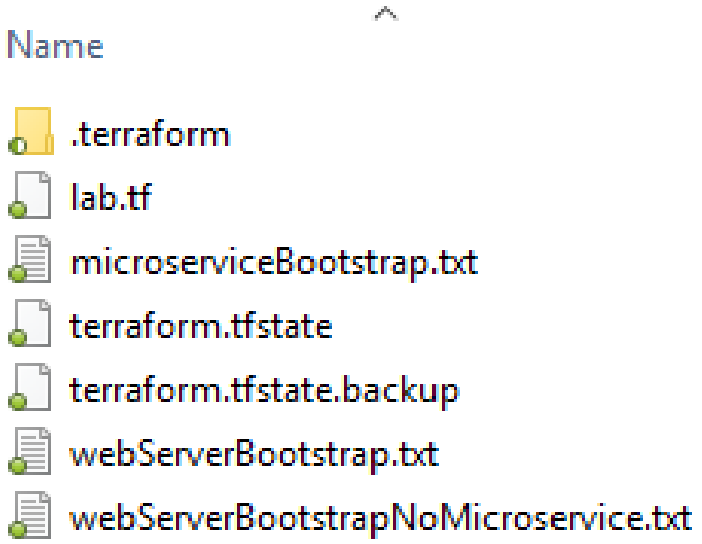
- Example Template Reuse
- Sample source

```
variable "aws_key" {}
variable "aws_secret_key" {}

module "aws-vpc" {
  source  = "../../terraform-modules/aws-vpc"
  aws_key = "${var.aws_key}"
  aws_secret_key = "${var.aws_secret_key}"
  vpc_name = "TerraformVPC"
}

output "VpcId" {
    value = "${module.aws-vpc.VpcId}"
}
```

©2018 Derek C. Ashmore, All Rights Reserved

Typical Project Structure





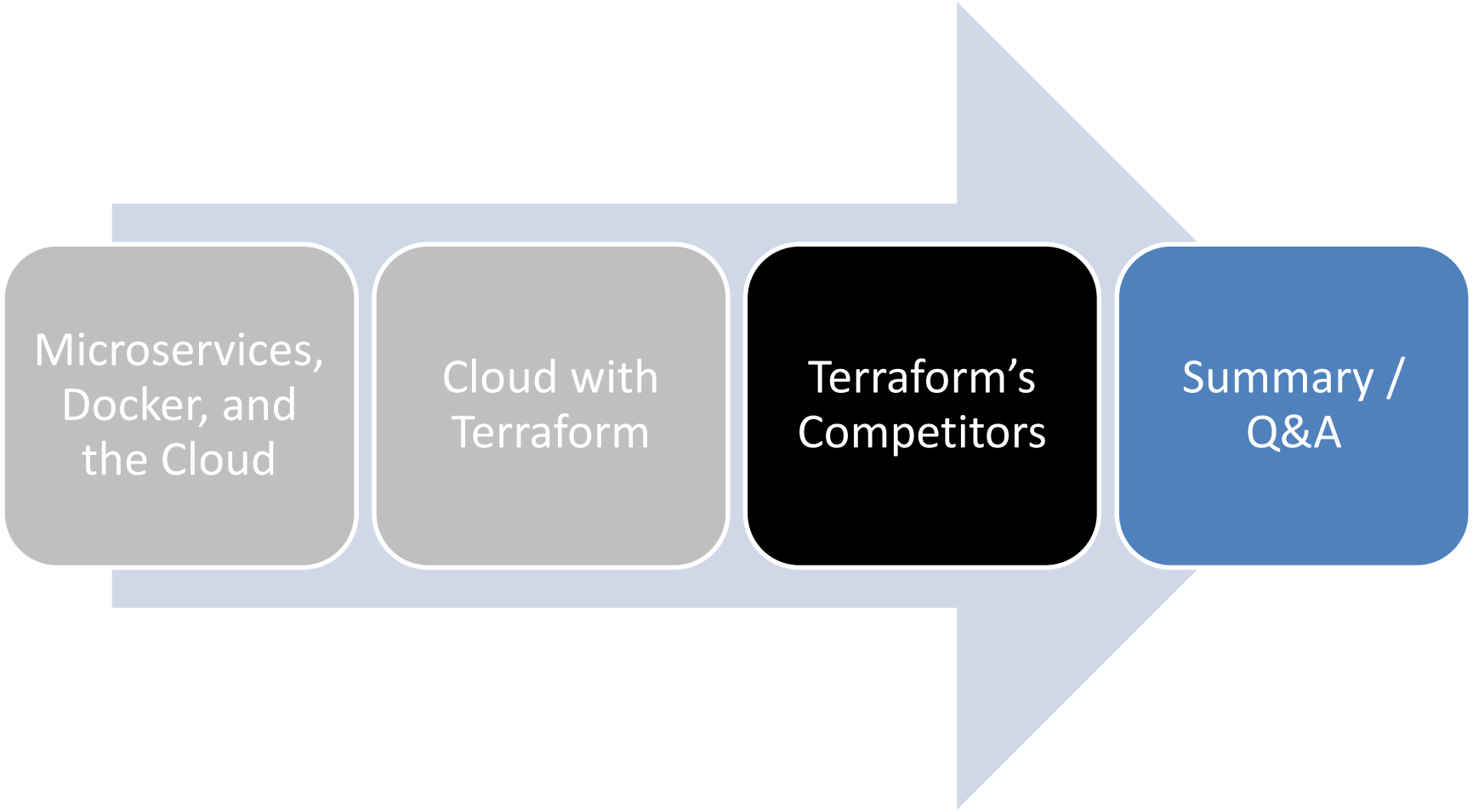
©2018 Derek C. Ashmore, All Rights Reserved

Terraform State

- Terraform stores state
  - Local file `terraform.tfstate`
- Teams need to manage state centrally
  - Terraform Backends
    - Locks so that only one person at a time can update
    - Remote storage
      - S3, Azure containers, Google cloud storage, etc.

©2018 Derek C. Ashmore, All Rights Reserved

Agenda



©2018 Derek C. Ashmore, All Rights Reserved

Terraform vs. Ansible/Chef

- Terraform designed for infrastructure
  - Not designed for configuration management
  - Terraform deploys images
    - Not good at maintaining what's on those images
- If deployments update existing VMs
  - You need Ansible, Chef, or Puppet
- If deployments are "new" VMs
  - Terraform can handle deployments too

©2018 Derek C. Ashmore, All Rights Reserved

Paradigm Shift

- Deployment as new infrastructure
  - New version → new VMs
    - Software versions baked into images
  - Advantages
    - Facilitates Canary Deployments
      - Route53 Routing Policies
    - Go-live operation has less risk
      - Deploy/Backout is just a load balancer switch
  - Disadvantages
    - More moving parts
    - Impossible to do manually

©2018 Derek C. Ashmore, All Rights Reserved

Terraform vs CloudFormation

Terraform

- Scripting skills translate to Azure, Google Cloud, etc.
- Less verbose (>50%)
- Data Lookups
- Custom Plug-ins possible
- Active Community Support

CloudFormation

- Quicker to follow AWS enhancements
- GUI support
- Automatic centralized state
- Vendor Support

©2018 Derek C. Ashmore, All Rights Reserved

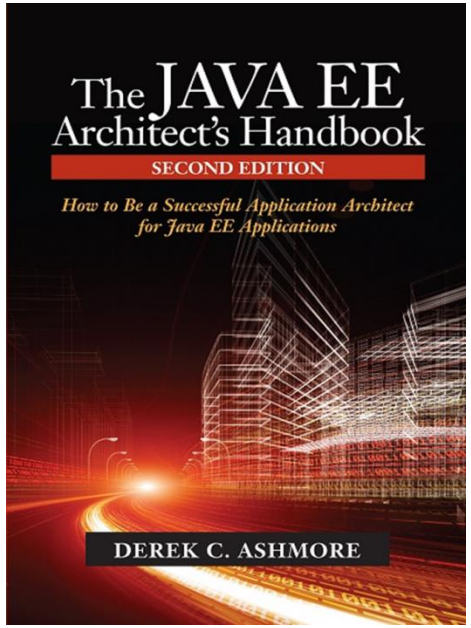
Further Reading

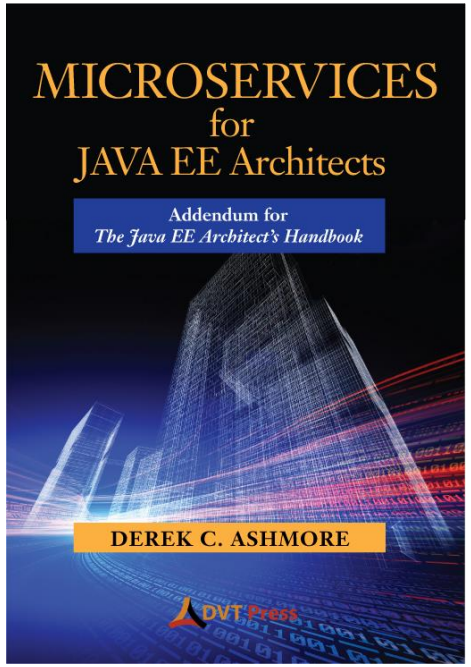
- This slide deck
  - http://www.slideshare.net/derekashmore
- The Gruntwork Blog
  - https://blog.gruntwork.io/
- The hands-on-lab code
  - https://github.com/Derek-Ashmore/terraform-hands-on-lab

©2018 Derek C. Ashmore, All Rights Reserved

Questions?

- Derek Ashmore:
  - Blog: www.derekashmore.com
  - LinkedIn: www.linkedin.com/in/derekashmore
    - Connect Invites from attendees welcome
  - Twitter: https://twitter.com/Derek_Ashmore
  - GitHub: https://github.com/Derek-Ashmore
  - Book: http://dvtpress.com/





ASPERITAS CONSULTING

©2018 Derek C. Ashmore, All Rights Reserved