Openly Deploying Open edX at MIT Open Learning

How We Use Open Source, Immutable Infrastructure for Running MITx Residential

Tobias Macey

DevOps Manager and Team Lead

@ MIT Open Learning

openlearning.mit.edu

Host of:

Podcast.__init__

Data Engineering Podcast

@TobiasMacey

github.com/blarghmatey

tmacey@mit.edu

Definition Time

Immutable Infrastructure: When you destroy and rebuild a server instead of updating it in place

Open Source Infrastructure: Releasing the code you use for production as open source for public reference (I gave a talk about this if you're curious)

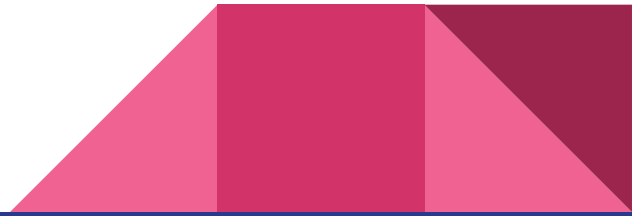
Configuration Drift: When a server doesn't match desired state because of multiple configuration updates/deployments

MITx Residential By The Numbers

- MITx Residential is the online course platform for MIT students
- 30 - 50 courses each semester
- Used by >90 MIT faculty and instructors
- Used by >4000 undergraduate students in MIT classes
- 91% of undergraduates have used the platform for coursework

The Cast Of Characters

- SaltStack for everything
  - Cloud provisioning
  - Configuration management
  - Reactive automation
- Vault for secrets
  - Integrated with Saltstack
  - Dynamic credentials
- Consul for service discovery
  - DNS interface for ease of implementation
- Ansible for edX app installation
  - Reduces maintenance burden



What Is SaltStack?

And Why We Use It

It's an automation framework with batteries included.

I introduced it at MIT Open Learning because:

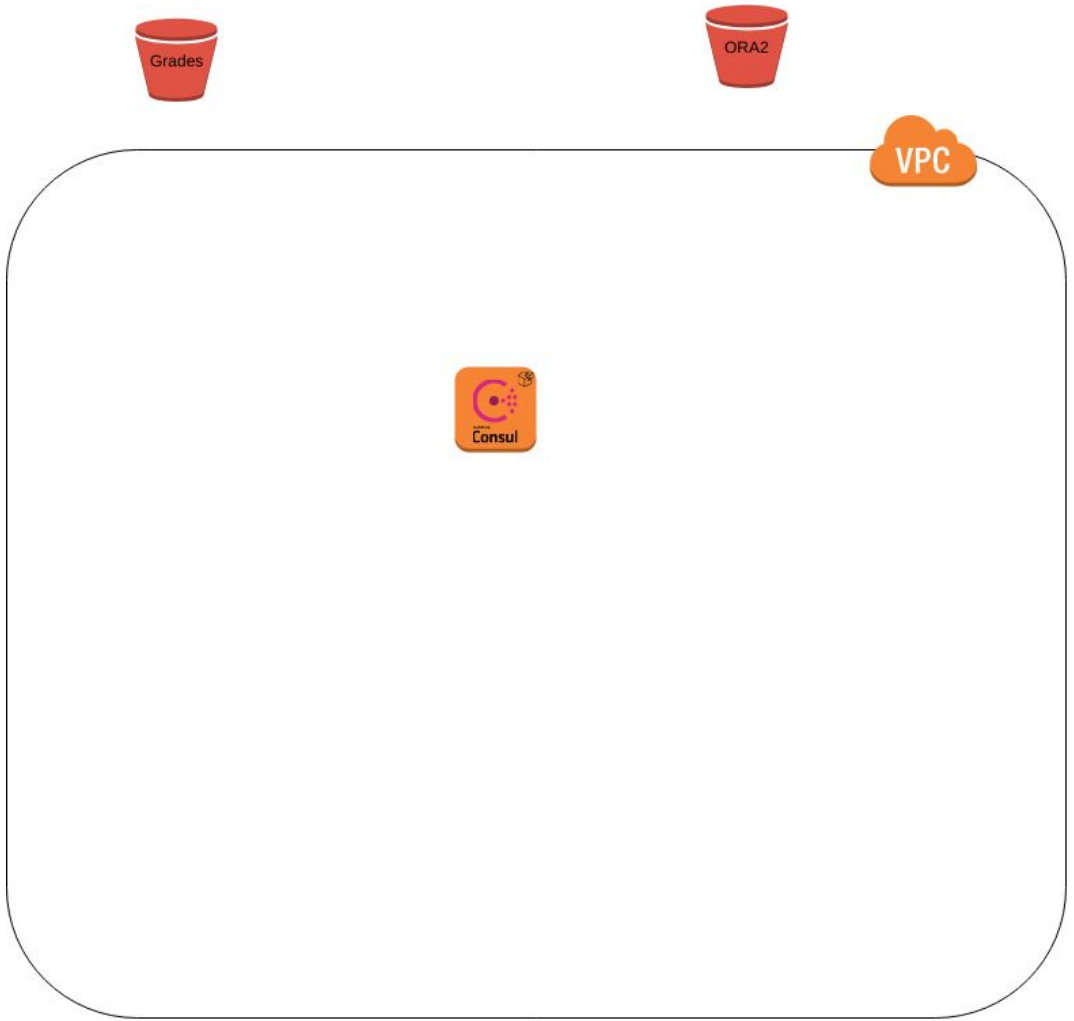
- It's modular and extensible
- Scales from one server to multiple data centers
- Event driven automation

The Layers

- Create VPC and S3 buckets

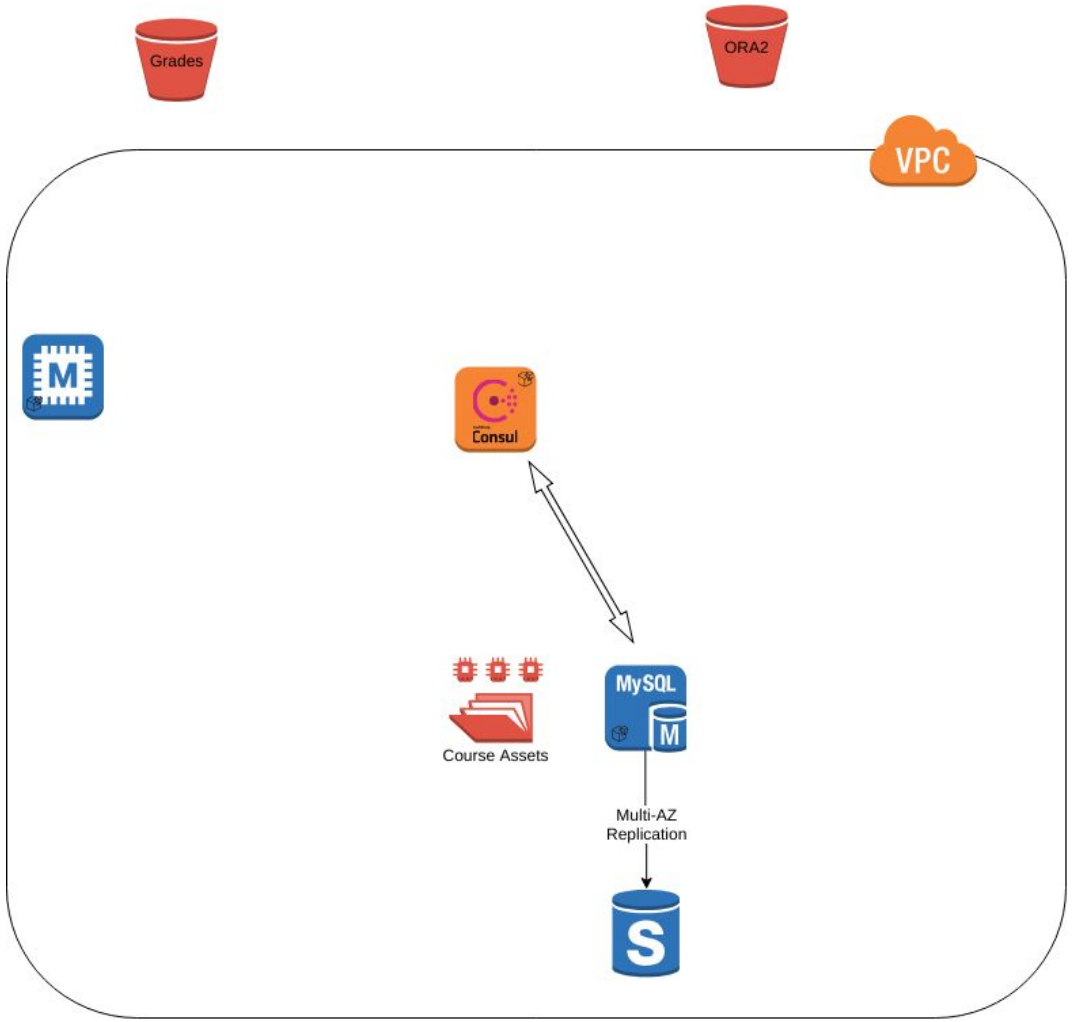
The Layers

- Create VPC and S3 buckets
- Deploy Consul cluster for service discovery (via DNS)



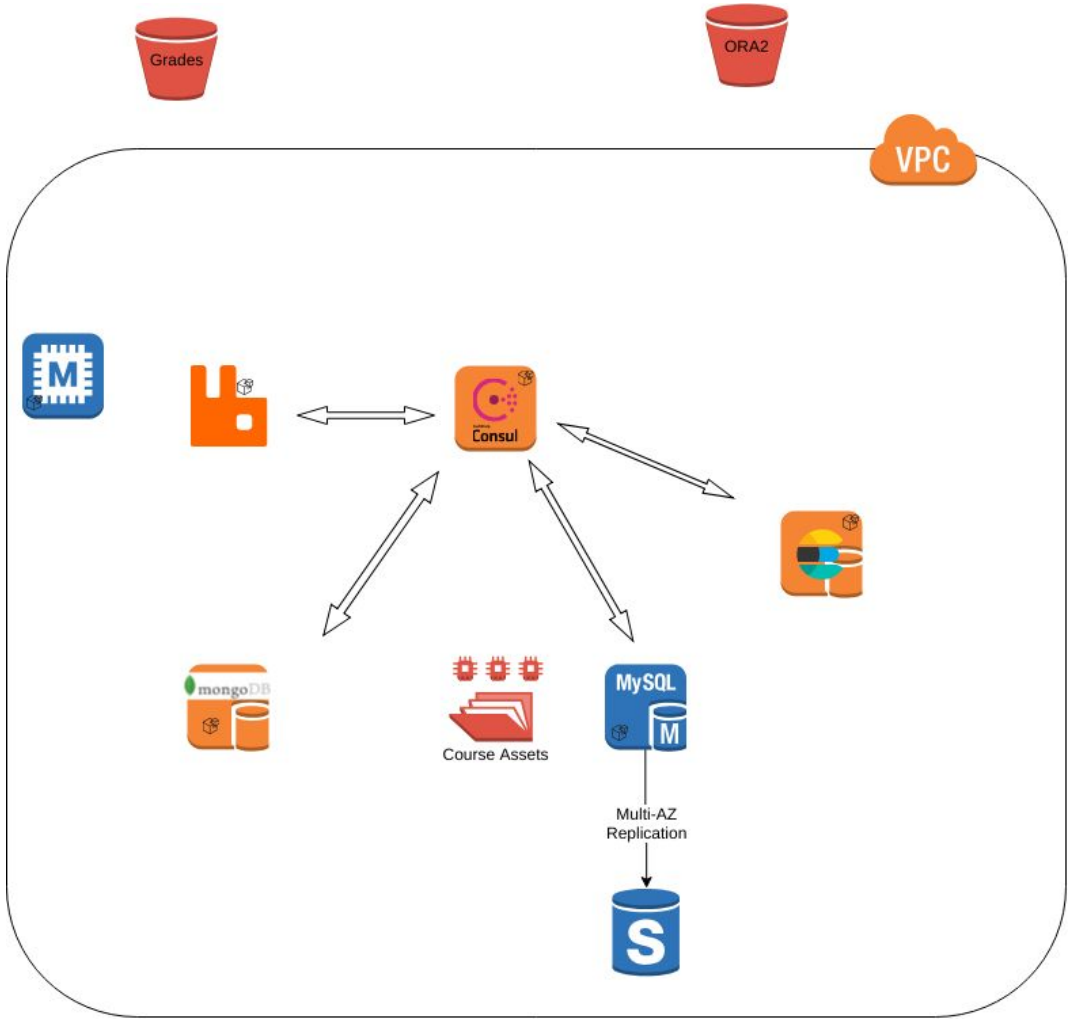
The Layers

- Create VPC and S3 buckets
- Deploy Consul cluster for service discovery (via DNS)
- Create RDS DB, EFS share for NFS, Memcached Elasticache cluster



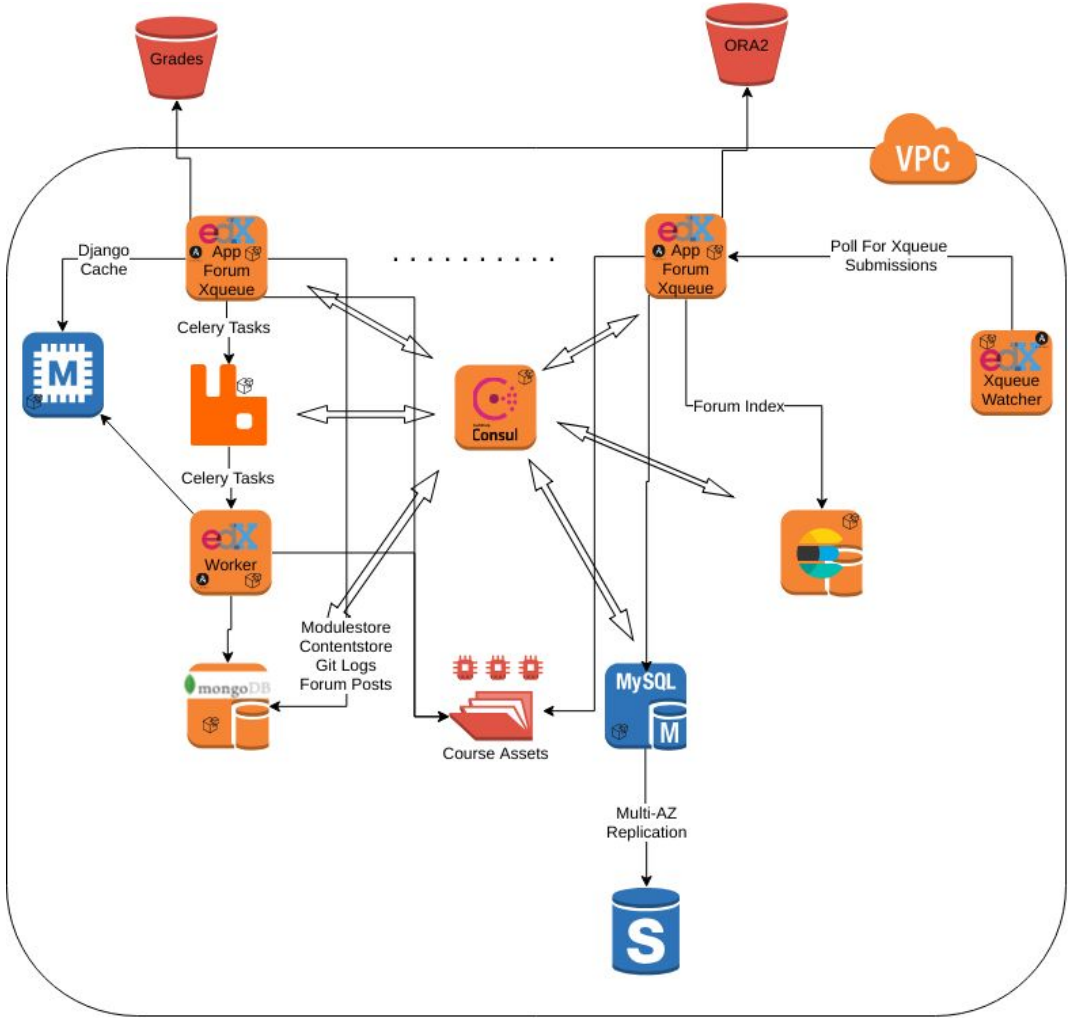
The Layers

- Create VPC and S3 buckets
- Deploy Consul cluster for service discovery (via DNS)
- Create RDS DB, EFS share for NFS, Memcached Elasticache cluster
- Deploy RabbitMQ, MongoDB, Elasticsearch clusters on EC2



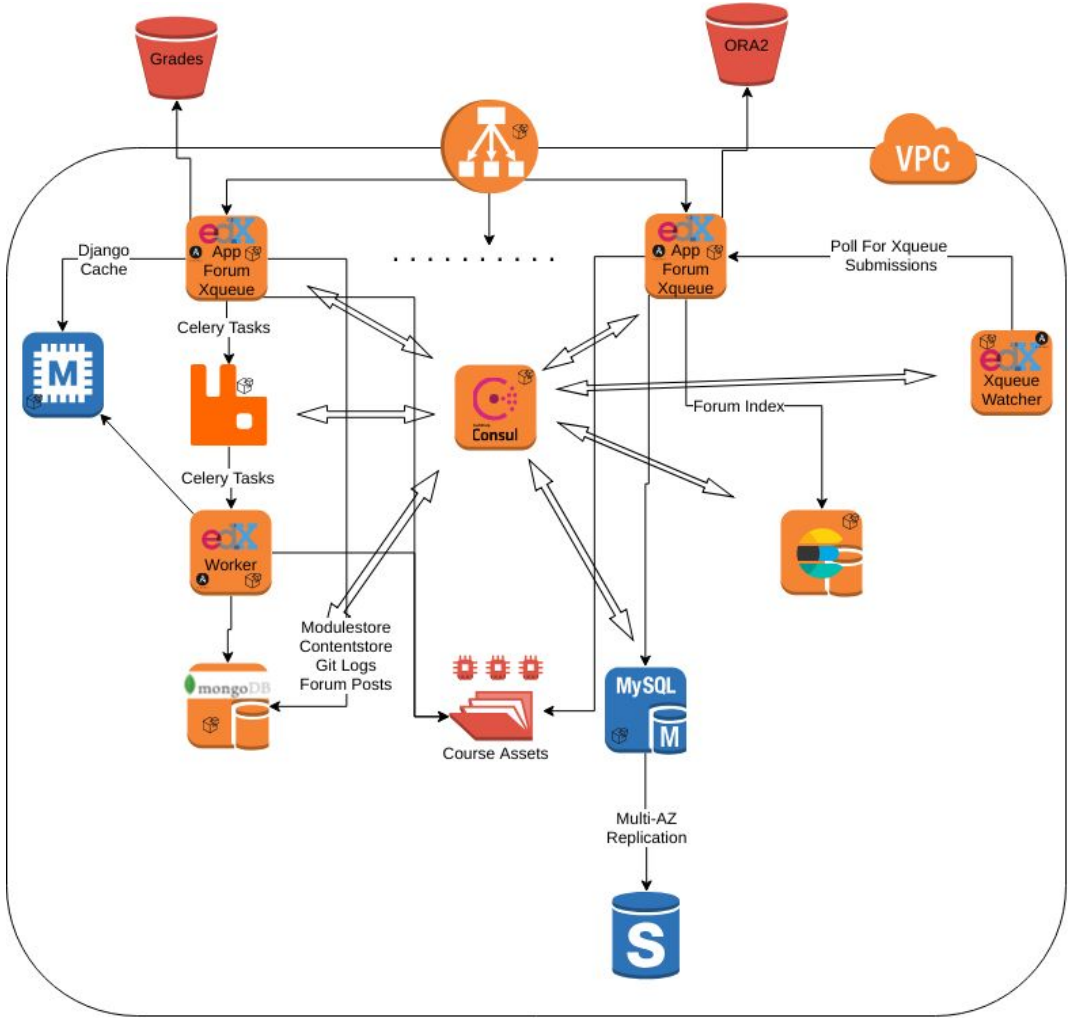
The Layers

- Create VPC and S3 buckets
- Deploy Consul cluster for service discovery (via DNS)
- Create RDS DB, EFS share for NFS, Memcached Elasticache cluster
- Deploy RabbitMQ, MongoDB, Elasticsearch clusters on EC2
- Deploy App, Worker, and Xqueue-Watcher instances



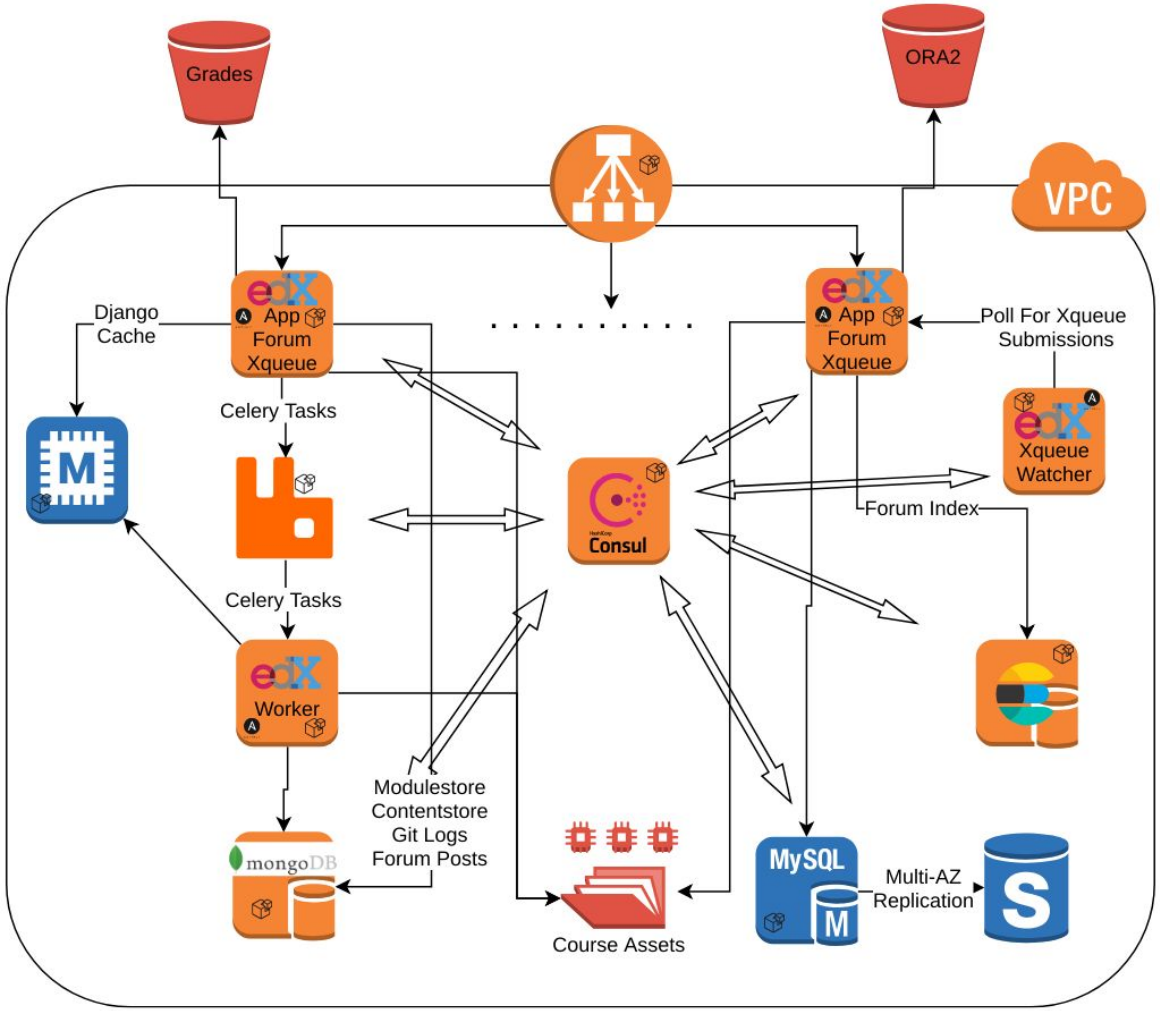
The Layers

- Create VPC and S3 buckets
- Deploy Consul cluster for service discovery (via DNS)
- Create RDS DB, EFS share for NFS, Memcached Elasticache cluster
- Deploy RabbitMQ, MongoDB, Elasticsearch clusters on EC2
- Deploy App, Worker, and Xqueue-Watcher instances
- Deploy ELB and attach app instances



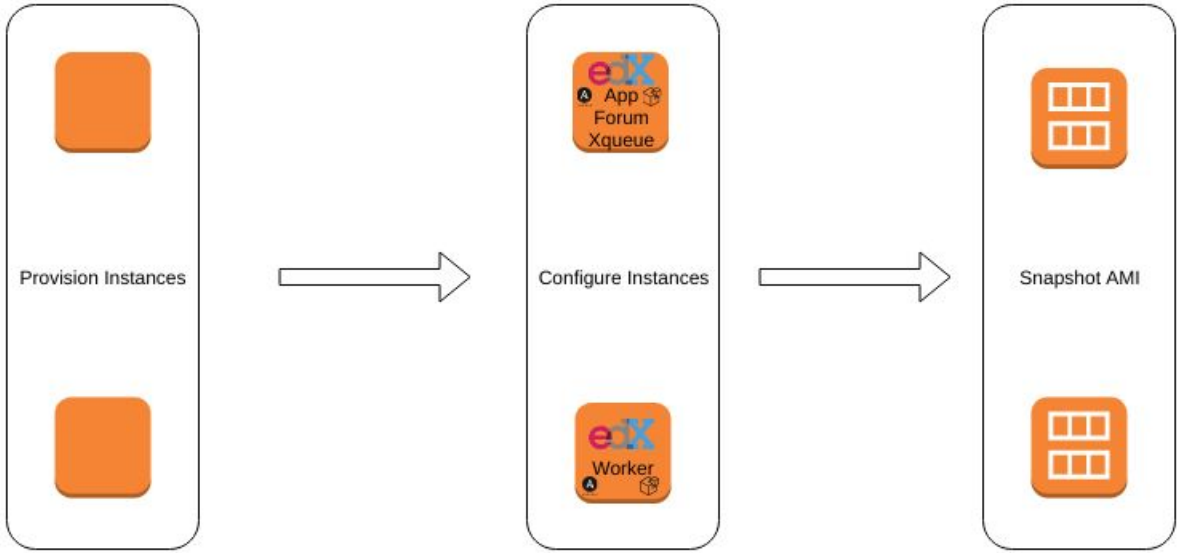
The Architecture

Stateless Apps ==
Easier Deployments



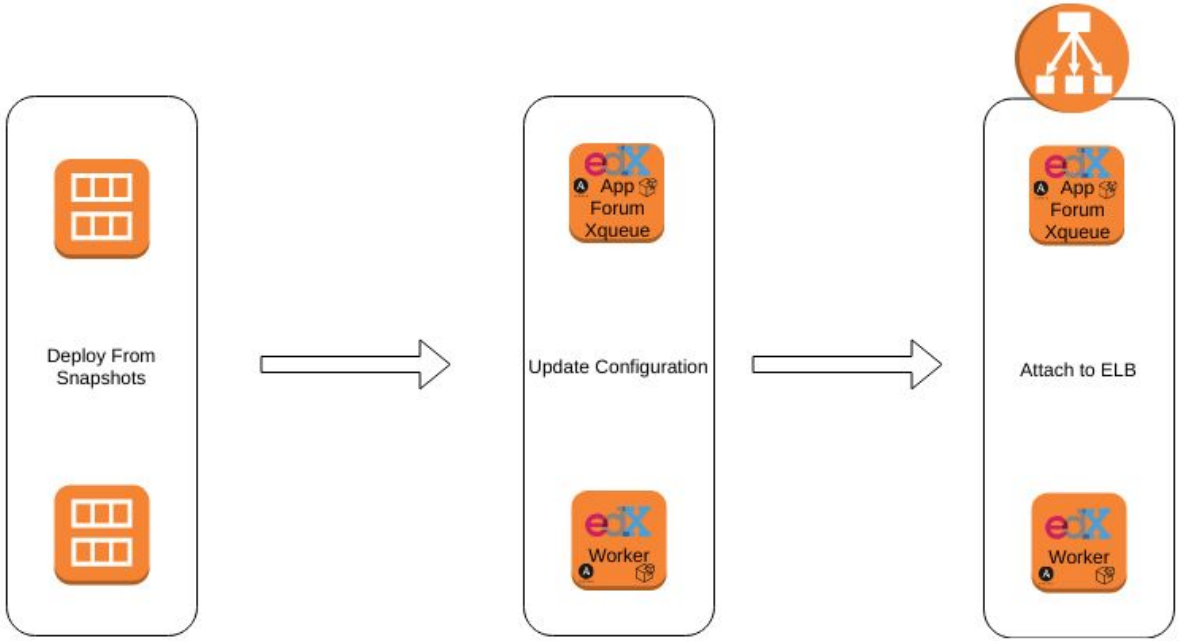
Deploying

- Provision bare servers
- Execute Ansible playbooks
- Snapshot as AMIs



Deploying

- Provision from AMI snapshots
- Update Configurations
- Attach to load balancer

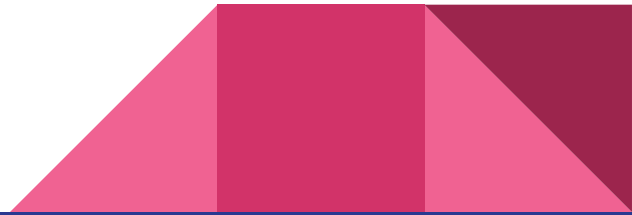


Version Upgrades

- Production data backed up and restored to QA environment
- Current production version runs alongside next target version
- Shared services, logically separated
  - RabbitMQ - separate vhosts
  - MySQL - separate schemas/databases
  - MongoDB - separate databases
  - Memcached - separate clusters
- Configuration files largely shared

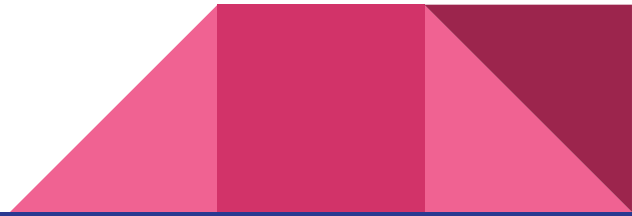
The Code

- Ansible vars managed by SaltStack
  - https://github.com/mitodl/salt-ops/tree/master/pillar/edx
- edX applications installed via Ansible playbooks
  - https://github.com/mitodl/salt-ops/blob/master/salt/edx/run_ansible.sls
- Infrastructure components built with corresponding Salt formulas
  - https://github.com/mitodl/rabbitmq-formula
  - https://github.com/mitodl/elasticsearch-formula
  - https://github.com/mitodl/consul-formula
  - https://github.com/mitodl/mongodb-formula



Where We Were

- Long-lived instances, updated in place
- Unreliable, unpredictable, slow deployments
- Limited scalability
- Error prone upgrade path

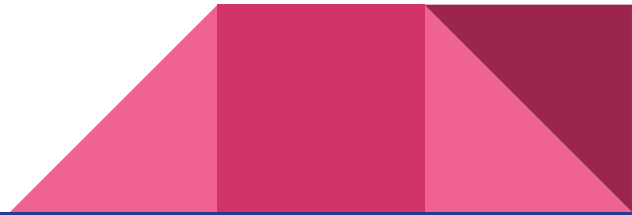


Where We Are Now

- Stable, scalable infrastructure
- Stateless, immutable application instances
- Multi-version testing environment for smooth upgrade path
- Modular configuration for customizing edX deployments
- Fast, reliable, and safe deployments for security patches, etc.

Where We Are Going Next

- Autoscaling
- Picking apart services more
- Rewrite Ansible roles as Salt formulas
- Reserved instances
- Replace ELB Classic with more flexible load balancing
- More detailed monitoring (metrics, performance data, memory profiling, better log aggregation, etc.)
- More comprehensive integration testing during deployments



Questions?