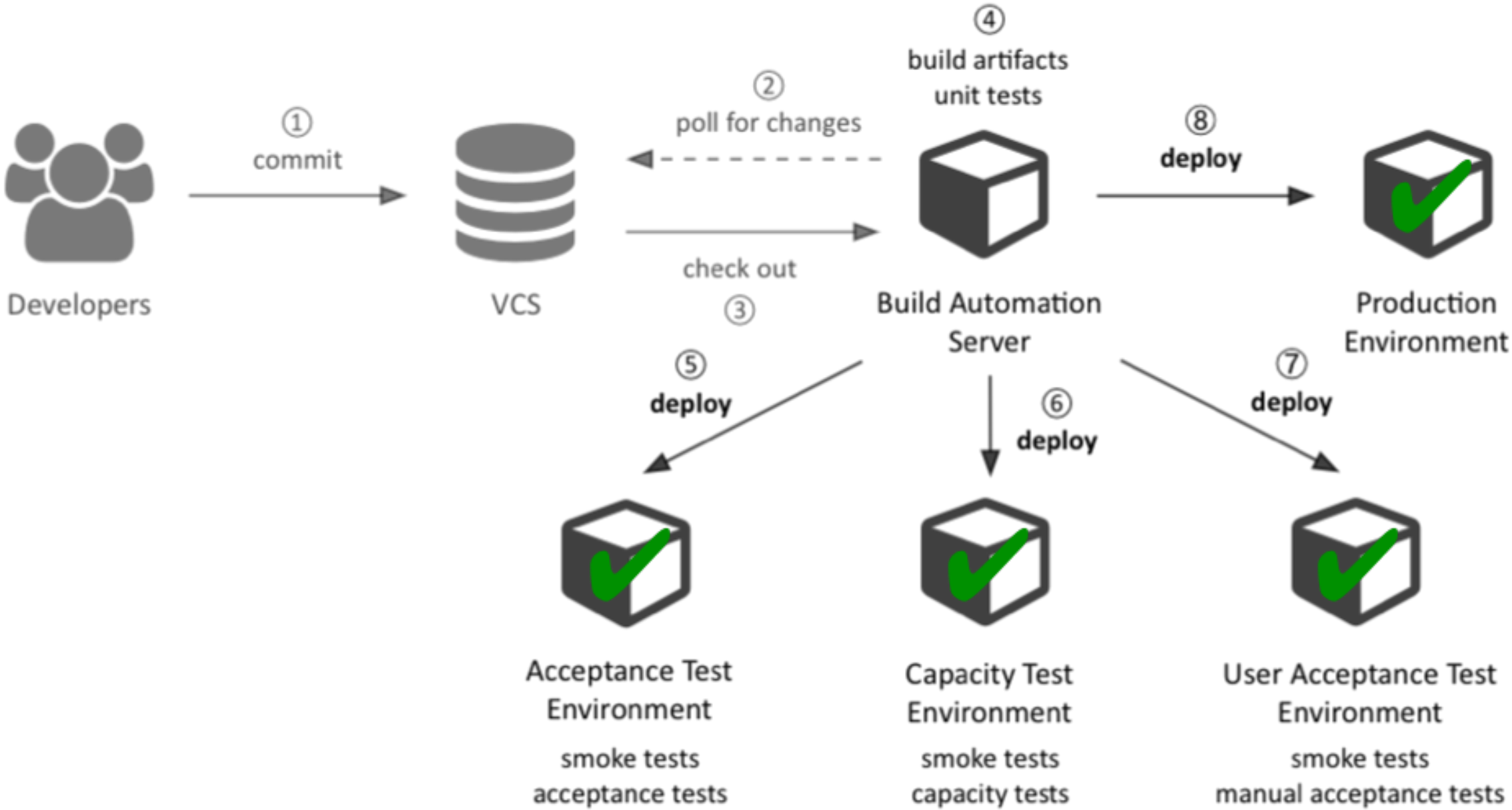
Midterm Presentation Schedule

- October 18th
  - Aurora, Bash, Sangam
- October 20th
  - Flash, Omega, CodeRing
- October 25th
  - Omni, Aviato, NPComplete

Docker, Continuous Integration, and Continuous Deployment

Combining Docker, Travis-CI, and Amazon Code Deploy


④
build artifacts
unit tests
②
poll for changes
①
commit
⑧
deploy
Developers
VCS
check out
③
Build Automation
Server
Production
Environment
⑤
deploy
⑥
deploy
⑦
deploy
Acceptance Test
Environment
smoke tests
acceptance tests
Capacity Test
Environment
smoke tests
capacity tests
User Acceptance Test
Environment
smoke tests
manual acceptance tests

Docker and Travis-CI

- Integration of Docker with Travis-CI is straightforward if you are doing things thoughtfully
  - Using a Dockerfile to define how to build your images.
  - Pushing images to Docker Hub for Travis to retrieve them.
  - Using Docker Compose to orchestrate multiple inter-dependent containers
- You can use Travis-CI to push your built images back to Docker Hub as well.
  - Your CD system can retrieve them from Docker Hub for deployment

https://docs.travis-ci.com/user/docker/

Example .travis-yml file for running an image

```
sudo: required

language: ruby

services:
  - docker

before_install:
- docker pull carlad/sinatra
- docker run -d -p 127.0.0.1:80:4567 carlad/sinatra /bin/sh -c "cd /root/sinatra; bundle exec foreman start;"
- docker ps -a
- docker run carlad/sinatra /bin/sh -c "cd /root/sinatra; bundle exec rake test"

script:
- bundle exec rake test
```

Example .travis.yml for a docker build to create an image

```
sudo: required

language: ruby

services:
  - docker

before_install:
  - docker build -t carlad/sinatra .
  - docker run -d -p 127.0.0.1:80:4567 carlad/sinatra /bin/sh -c "cd /root/sinatra; bundle exec foreman start;"
  - docker ps -a
  - docker run carlad/sinatra /bin/sh -c "cd /root/sinatra; bundle exec rake test"

script:
  - bundle exec rake test
```

You can also use built-in docker-compose

Example .travis.yml file for pushing an image

```
travis env set DOCKER_EMAIL me@example.com
travis env set DOCKER_USERNAME myusername
travis env set DOCKER_PASSWORD secretsecret
```

Within your `.travis.yml` prior to attempting a `docker push` or perhaps before `docker pull` of a private image, e.g.:

```
docker login -e="$DOCKER_EMAIL" -u="$DOCKER_USERNAME" -p="$DOCKER_PASSWORD"
```

And then “docker push”

Set your username and password as Travis-CI environment variables

In Other Words...

- Docker support on Travis-CI is nothing special.
- There is built-in support for Docker
- But you just use it as you would otherwise.

Amazon is a different story.

Docker and Amazon Container Service

"Amazon ECS makes building and running containerized applications simple, but how that happens is what makes Amazon ECS interesting. " –Werner Vogels, Amazon CTO

http://www.allthingsdistributed.com/2015/07/under-the-hood-of-the-amazon-ec2-container-service.html

Docker on VM-Based Clouds

- You can run Docker on Hypervisor VMs
  - Docker containers inherit the parent VM's overhead but also increased sandboxing.
    - You don't share the real kernel with another user
  - But you can map many microservices to a few VMs.
  - This is much better than one VM per microservice
- Infrastructure as a Service uses hypervisor-based VMs
  - More secure sandboxing of clients from each other

If you are Amazon CTO Werner Vogels, are you worried about Docker? How can AWS add value?

http://www.allthingsdistributed.com/

The Problem Is Scale

- Running a few containers is easy.
- But a microservice architecture assumes lots of different services, each replicated many times.
- Vogels: this is a cluster management problem.
  - Configuration management
  - Service discovery
  - Scheduling
  - Monitoring systems
- And Amazon knows scale

Traditional Cluster Management

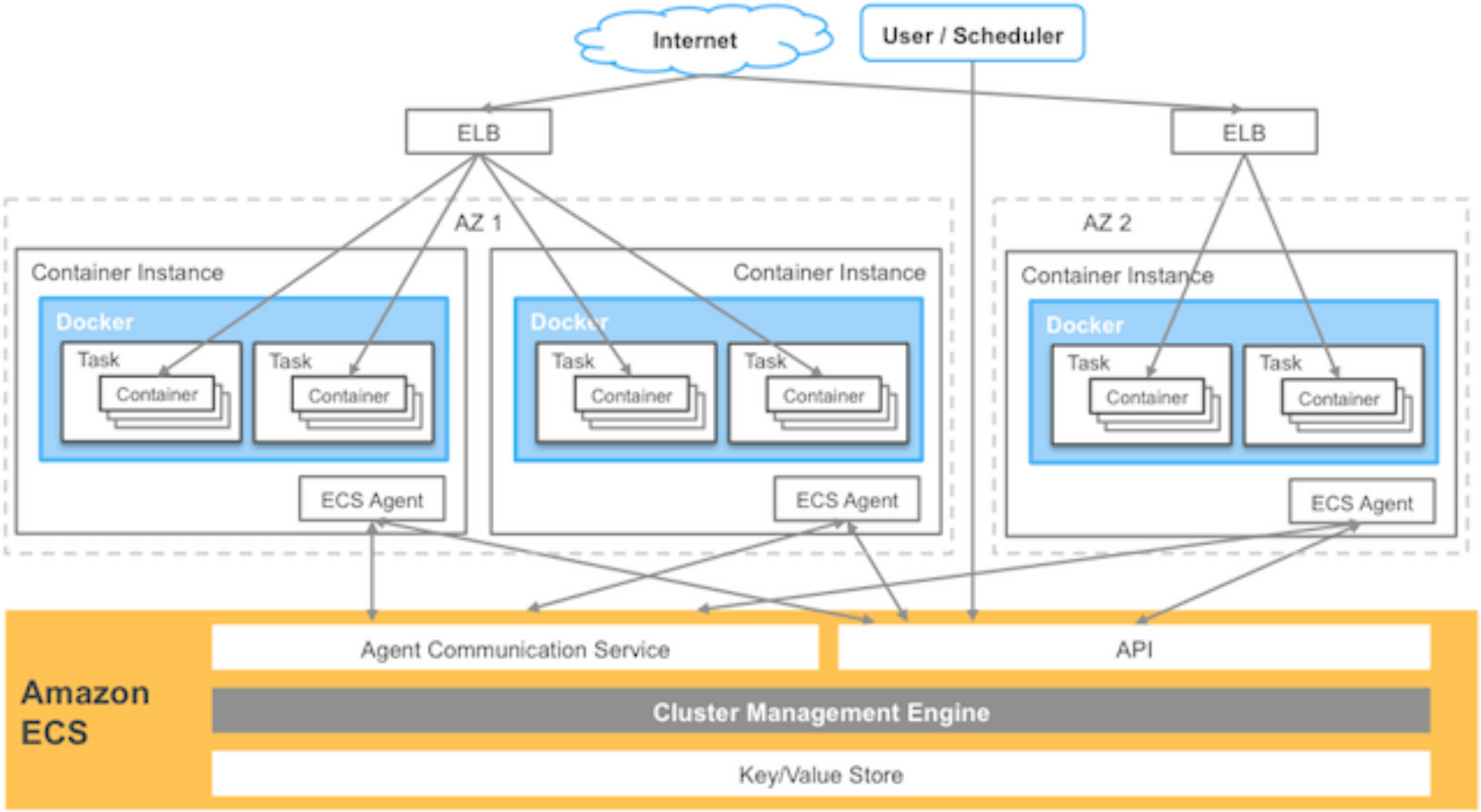
- Example Cluster: IU's Karst System
- Dedicated to a single application or would be statically partitioned to accommodate multiple users.
- Jobs go to job queues to ensure fairness and increased cluster utilization.
  - Scheduling and queuing problem
- Good for running scientific applications but not a good fit for microservices
  - Not designed for long running applications
  - Not designed to dynamically scale
  - Users need their own internal scheduling systems

We have lectures on this topic.

Requirements for Distributed Microservice Applications

- Reliable state management of your deployment and the underlying cluster.
  - EC2 instances in the clusters, tasks running on the EC2 instances, containers that make up a task, and resources available or occupied (e.g., networks ports, memory, CPU, etc).
  - This is a service provided by a **Cluster Manager**
- Flexible scheduling
  - Choose the resource scheduler that is right for your application
- It is best practice to decouple scheduling and resource management
  - Examples: Torque/MOAB and Apache Mesos

http://www.allthingsdistributed.com/2015/04/state-management-and-scheduling-with-ecs.html



"The ECS agent allows Amazon ECS to communicate with the EC2 instances in the cluster to start, stop, and monitor containers as requested by a user or scheduler. "

"The core of Amazon ECS is the cluster manager, a backend service that handles the tasks of cluster coordination and state management. On top of the cluster manager sits various schedulers."

Apache Mesos?

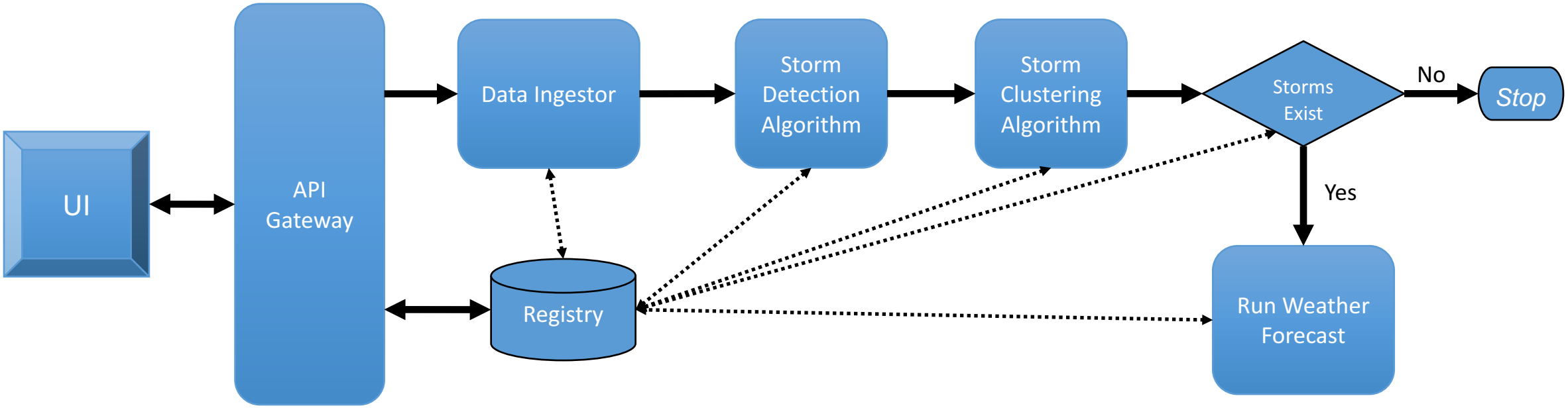
Amazon ECS State Management

- ”State” is what your system looks like at a given time
  - EC2 instances in the clusters, tasks running on the EC2 instances, containers that make up a task, and resources available or occupied (e.g., networks ports, memory, CPU, etc).
- State needs to be stored somewhere,
- Modern cluster managers use a key/value store.
  - Example: Apache Zookeeper ← Future Lecture
- Key/Value store is a the single source of truth about the state of your system.
  - It is a distributed system itself.
  - It needs to scalably and fault tolerantly handle transactions
  - Paxos and Zab are example underlying protocols
- Amazon exposes its cluster manager to users through an API

Microservice Scheduling

- Schedulers start and stop containers
  - How, when, and where?
- Schedulers are decoupled from cluster managers
  - Interact through the cluster manager API
  - Apache Mesos is the prototype for cloud schedulers
- You can use the scheduler that is appropriate for your application
- **What are your scheduling tasks?**

What Do You Need to Schedule?



Example Microservice Scheduling Tasks

- Deploy an entire new suite of microservices
- Update all instances of an existing microservice
- Restart a failed microservice
- Reroute work if away from a failed microservice
- Decide if you need to scale up or scale down your deployments.
- Internal scheduling of user-requested work
  - This is the traditional cluster scheduling problem
  - You should see this in your capacity testing for Project Milestone 2

Steps for Using ECS

Create a Task Definition	• This describes a deployment of one or more services in containers.
Configure a cluster	• A grouping of EC2 instances • Autoscaling groups
Create a Task with an associated Scheduler	• Task: Instantiation of your Task Definition • Scheduler: decides when and where to create task

ECS Task Definition JSON

- Describes one or more containers that form your application.
- Similar to a Dockerfile
  - Linking connected images
  - Specifying exposed ports and port mappings.
  - Memory: maximum MiB available to the container
  - CPU: minimum CPU available to the container
- *Family*: the name of the task
- *Essential:* if the container fails, the whole task fails

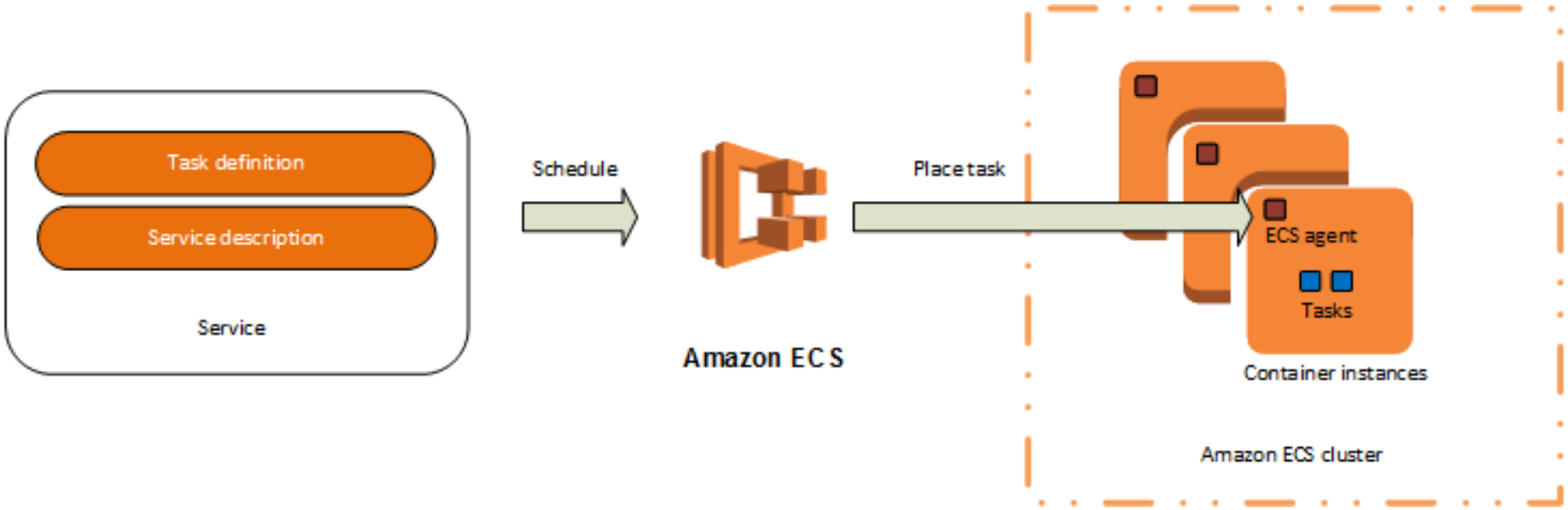
```
{
  "containerDefinitions": [
    {
      "name": "wordpress",
      "links": [
        "mysql"
      ],
      "image": "wordpress",
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "memory": 500,
      "cpu": 10
    },
    {
      "environment": [
        {
          "name": "MYSQL_ROOT_PASSWORD",
          "value": "password"
        }
      ],
      "name": "mysql",
      "image": "mysql",
      "cpu": 10,
      "memory": 500,
      "essential": true
    }
  ],
  "family": "hello_world"
}
```

Thought Exercise

How many Task Definitions do you need for your Project Milestones?

Tasks and Scheduling

- Task: the instantiation of a task definition on a container instance within a cluster.
  - You can specify the number of tasks that will run on your cluster.
- The Amazon ECS **task scheduler** places tasks on container instances.
- There are several different scheduling options available.
  - *Service* scheduler runs and maintains a specified number of tasks simultaneously.
  - *RunTask* scheduler is good for running batch processing tasks



Service Scheduler

- Use for long running stateless services and applications.
- Ensures that the specified number of tasks are constantly running and reschedules tasks when a task fails.
- Optionally also makes sure that tasks are registered against an Elastic Load Balancing load balancer.

RunTask

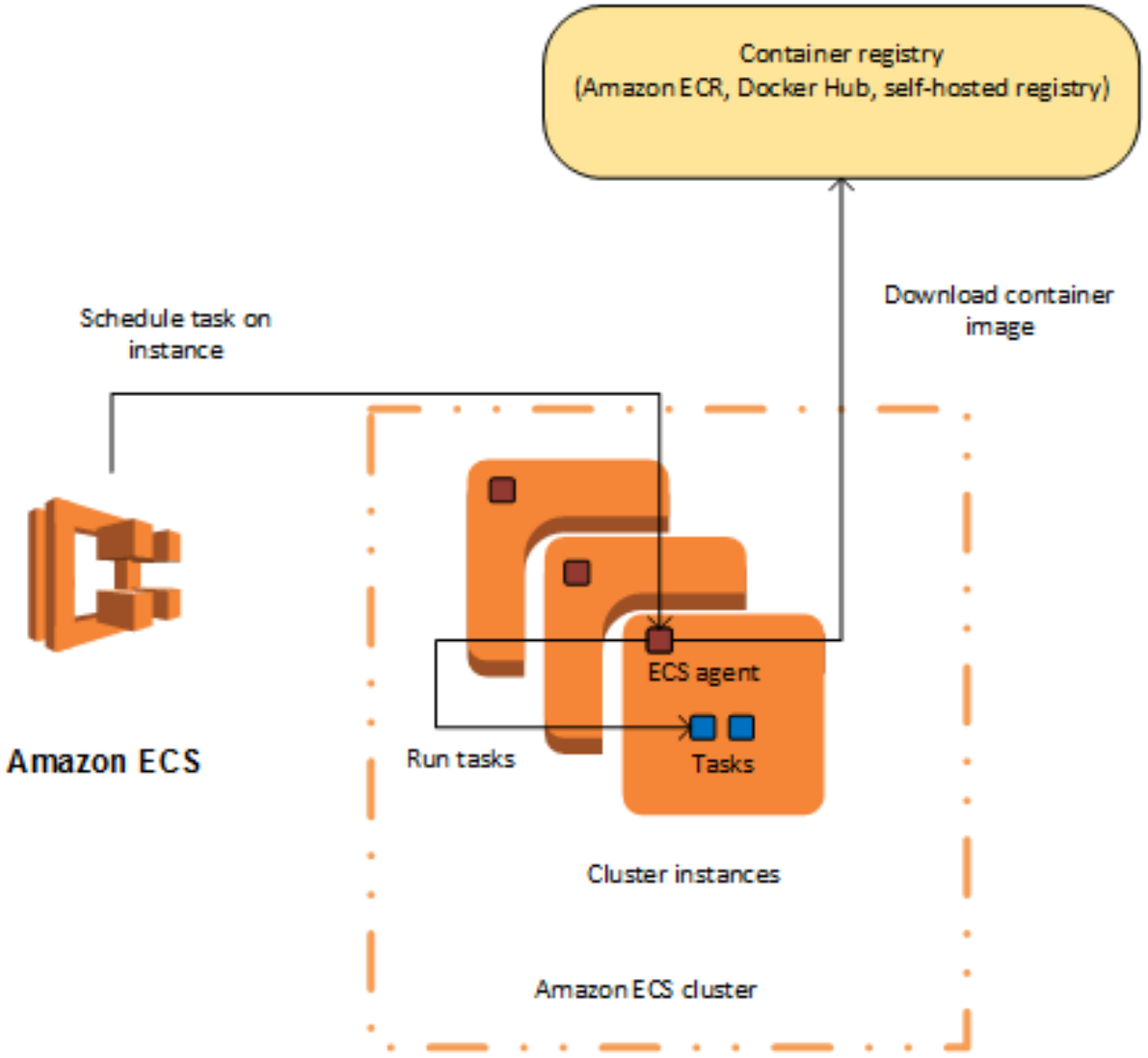
- Suited for processes such as batch jobs that perform work and then stop.
- Randomly distributes tasks across a cluster
- Tries to minimize the chances that a single instance on your cluster will get a disproportionate number of tasks.

ECS Clusters

- Tasks run on clusters
  - Logical grouping of EC2 instances.
- Amazon ECS downloads your container images from a registry and runs those images on the container instances within the cluster.
- You can scale your clusters using Amazon's Autoscaling features.
- You can also add and remove EC2 instances "manually"

ECS Container Agent

- Runs on each instance within an Amazon ECS cluster.
- Sends information about the instance's current running tasks and resource utilization to Amazon ECS
- Starts and stops tasks whenever it receives a request from Amazon ECS
  - You (via API call)
  - Scheduler



Apache Mesos and the Mesosphere

- Apache Mesos is an open source resource manager that sounds very similar to Amazon's resource manager
- Mesos works with plugin-schedulers
  - Marathon: runs long running tasks
  - Apache Aurora: also for scheduling persistent tasks
  - Chronos: batch scheduling
- The Mesosphere is the collection of Mesos plugins.
- We'll have a lecture about this in the future.
- Amazon tends to re:Invent.