SAND2016-3564C

Exceptional service in the national interest

Sandia National Laboratories

High-Performance Containerization Challenges

Colin Stapleton
ctstapl@sandia.gov

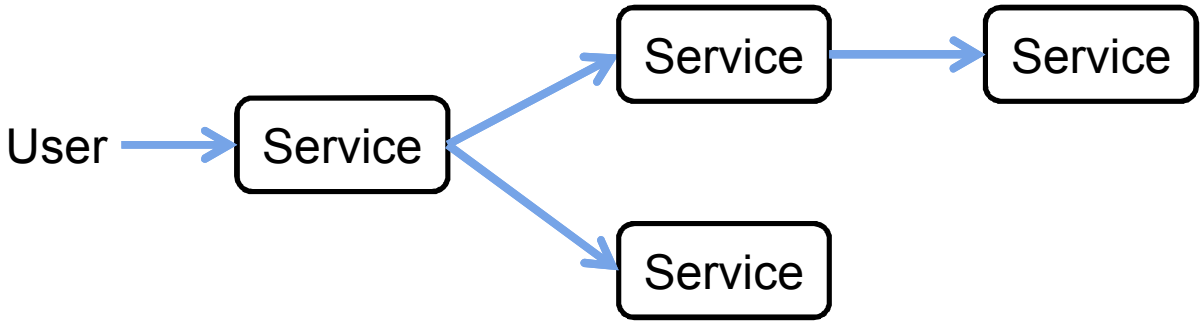
U.S. DEPARTMENT OF ENERGY
National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Overview

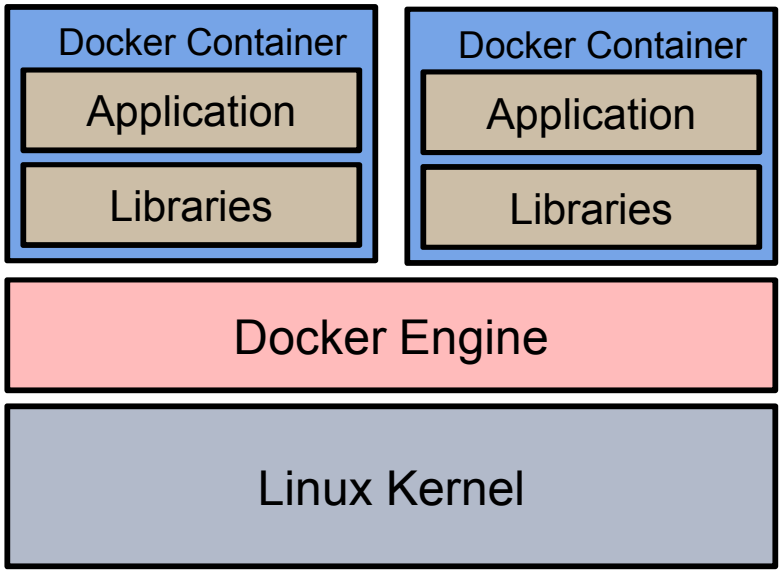
- Microservices intro
- Docker intro
- Kubernetes intro
- Containerization difficulties
  - Distributed architecture
  - Cloud environment

Microservice Software Design



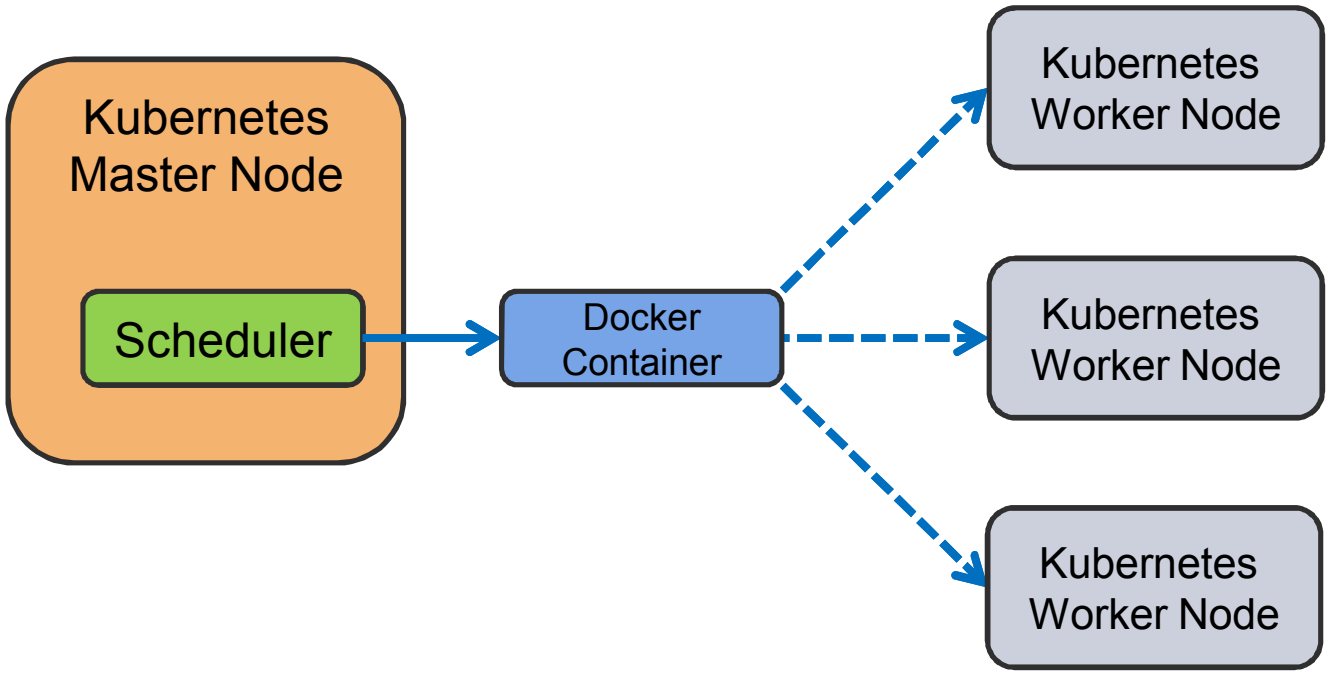
- Independent processes
  - Swap out easily
  - Can be developed by different groups
  - Single failure doesn't break entire system
  - Can use different technology for each service
- Easier for developers – less complicated components
- Language-agnostic messaging between services
- Distribute load across cluster of machines
  - Easier to scale both services and hardware
- Network connection between services adds complexity
- Large number of applications adds complexity

Use of Docker Containers



- Docker is like a “mini” virtual machine
  - Separate file system
  - Separate process space
  - Same kernel
- Put each service in separate container
- Combine application with dependencies
- Container images can be versioned and layered
- Images can be shared with other groups

Use of Kubernetes



- Schedules Docker containers on cluster of machines
  - Pick machine with correct hardware
  - Load-balancing
- Monitors and restarts containers
- Groups containers into logical “pods”

Potential Difficulties

Sandia National Laboratories

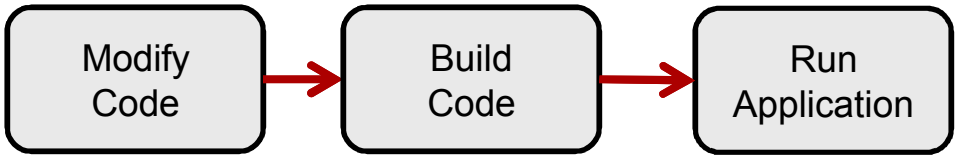
- Development workflow
- Debugging Inside Containers
- Separating applications from machines
- Application configuration
- Service discovery
- Avoiding race conditions
- Coordination between teams
- Logging
- Container deployment
- High-speed container startup

Development Workflow

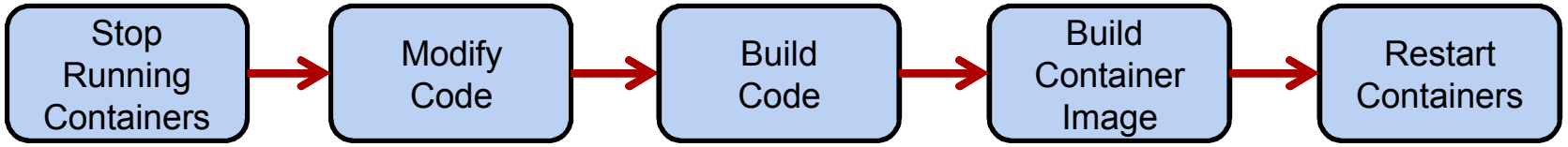
Sandia National Laboratories

Developing and deploying in different environments is a bad idea

Traditional Workflow



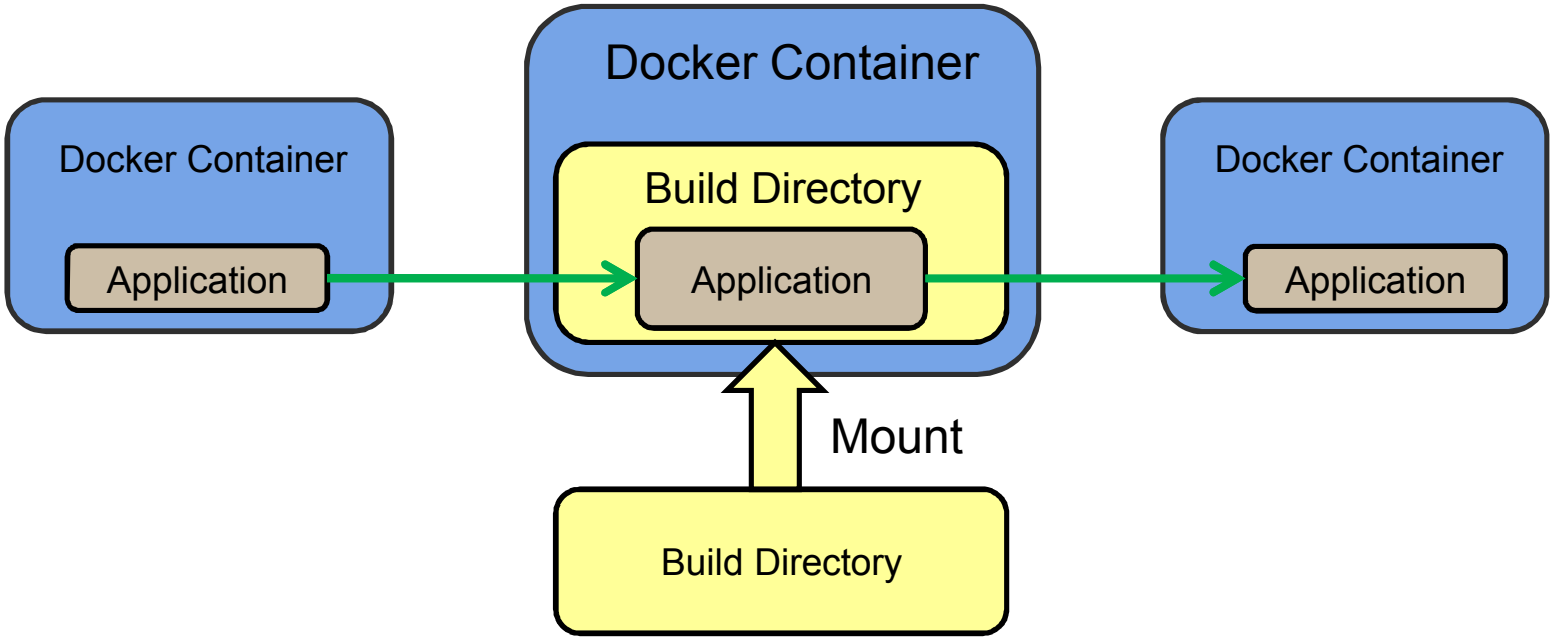
Container Workflow



Better to develop inside containers. However…

- Building container image can take seconds
- Starting containers can take seconds
- How can we make this process fast enough?

Development Workflow



Possible solution:

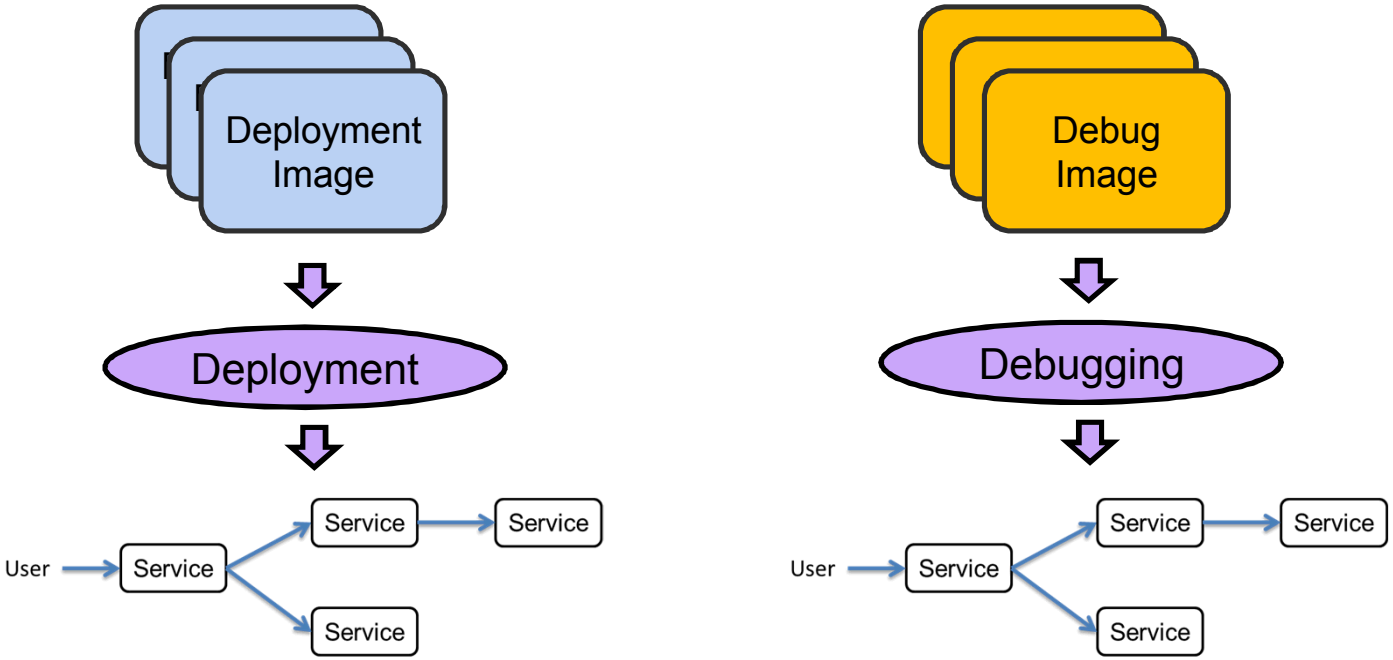
- Leave all containers running
- Mount development directory into container
- Rebuild and restart application inside container
- Applications auto-reconnect

Debugging Inside Containers

Sandia National Laboratories

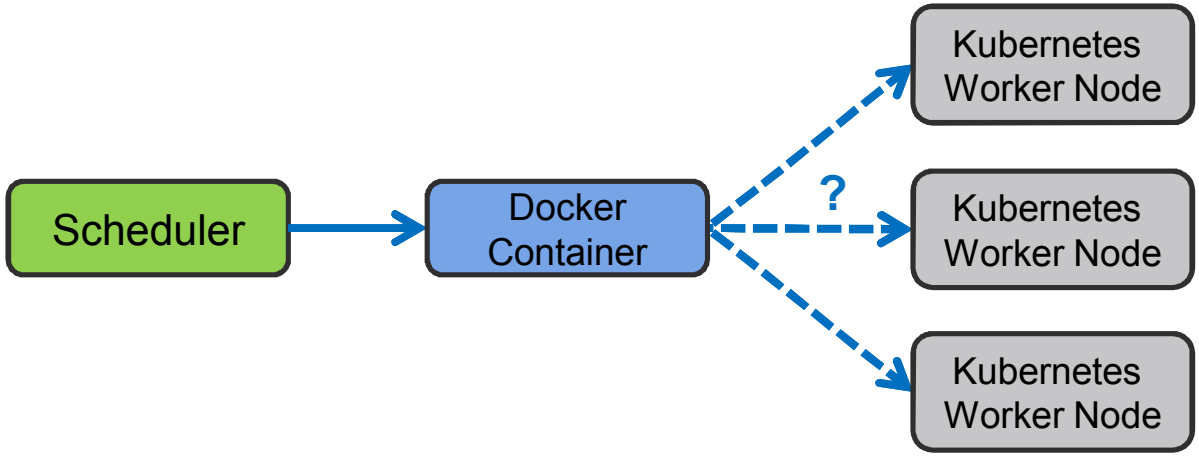
- The containers start up… and don't work. What do we do?
  - Examine log files
  - Examine configuration files
  - Think hard
- Is there a better way to see what is actually going on?

Debugging Inside Containers



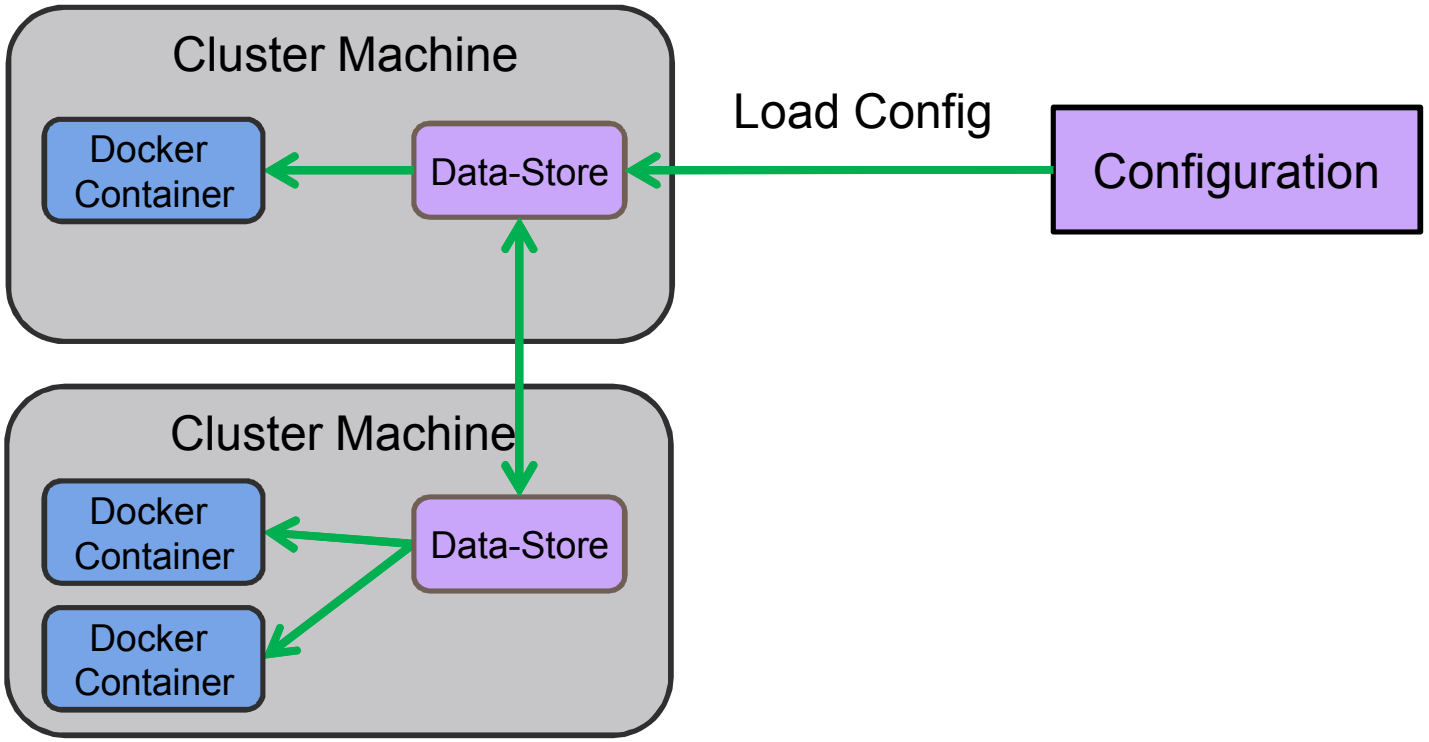
- Separate set of debug container images
  - Compiled with debug symbols
  - Contain debug tools
- Can easily swap out in Kubernetes config files

Separating Applications From Machines



- Application starts on an arbitrary machine
- All machines have standard setup
- No expectation of permanent files on machines
- How to pass configuration to an application in a cluster?
- How to ensure machine has correct resources for the application?

Application Configuration



- Store configuration in distributed data store
- Dynamically load config into apps on startup

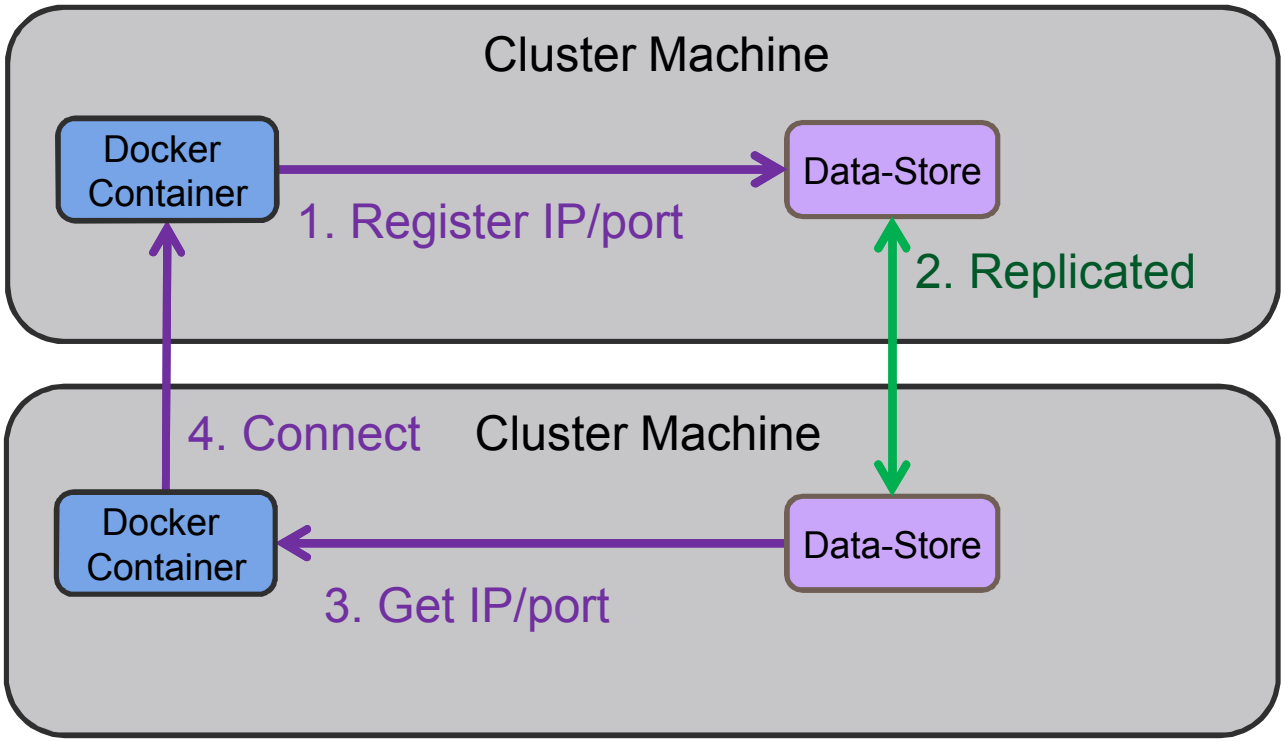
Service Discovery

Sandia National Laboratories

How do applications find each other in the cluster?

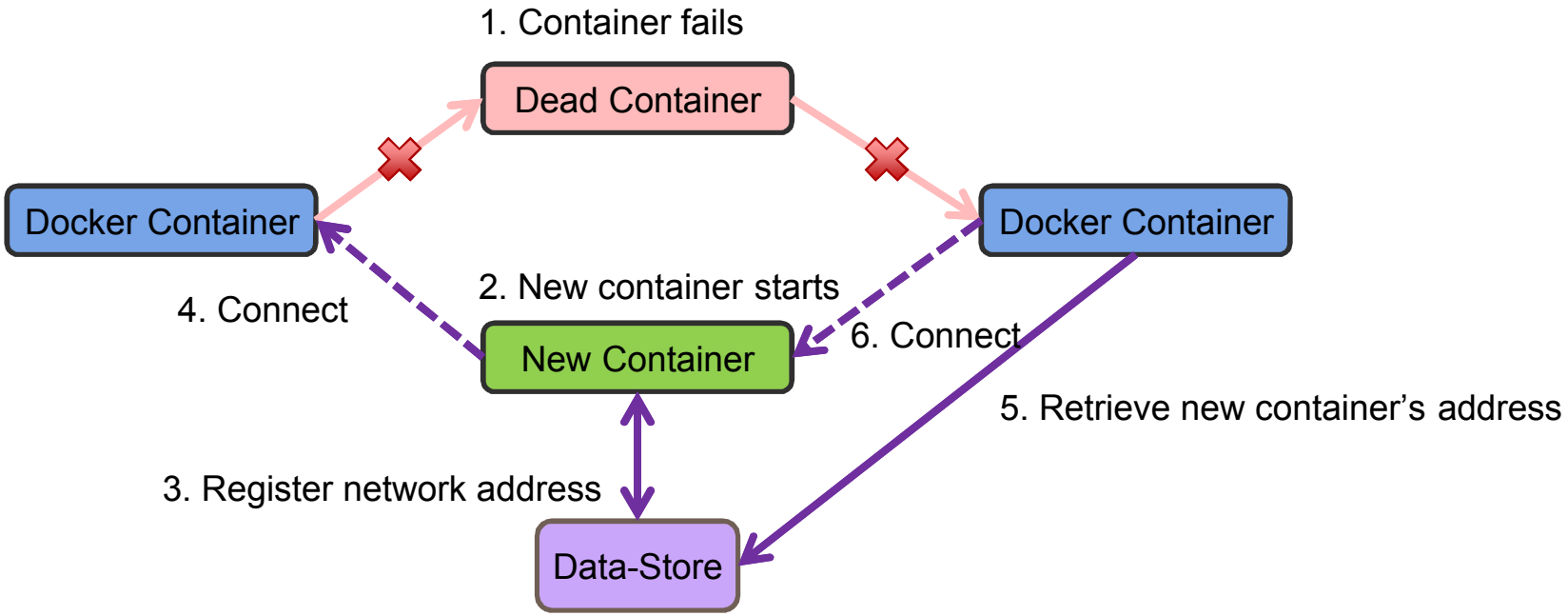
- We don’t know which machine an application will start on
- Machines all have identical setup
- Machines can restart at any time
- Applications can restart at any time on any machine

Service Discovery



- Applications find their network address on startup
  - Place address in distributed data-store
- Data-store is replicated across every machine
- Connecting application gets address from data-store

Avoiding Race Conditions



How does system respond when a service dies?

- Startup of containers and pods not in order
  - Containers must wait for incoming connections
- Services must auto-connect

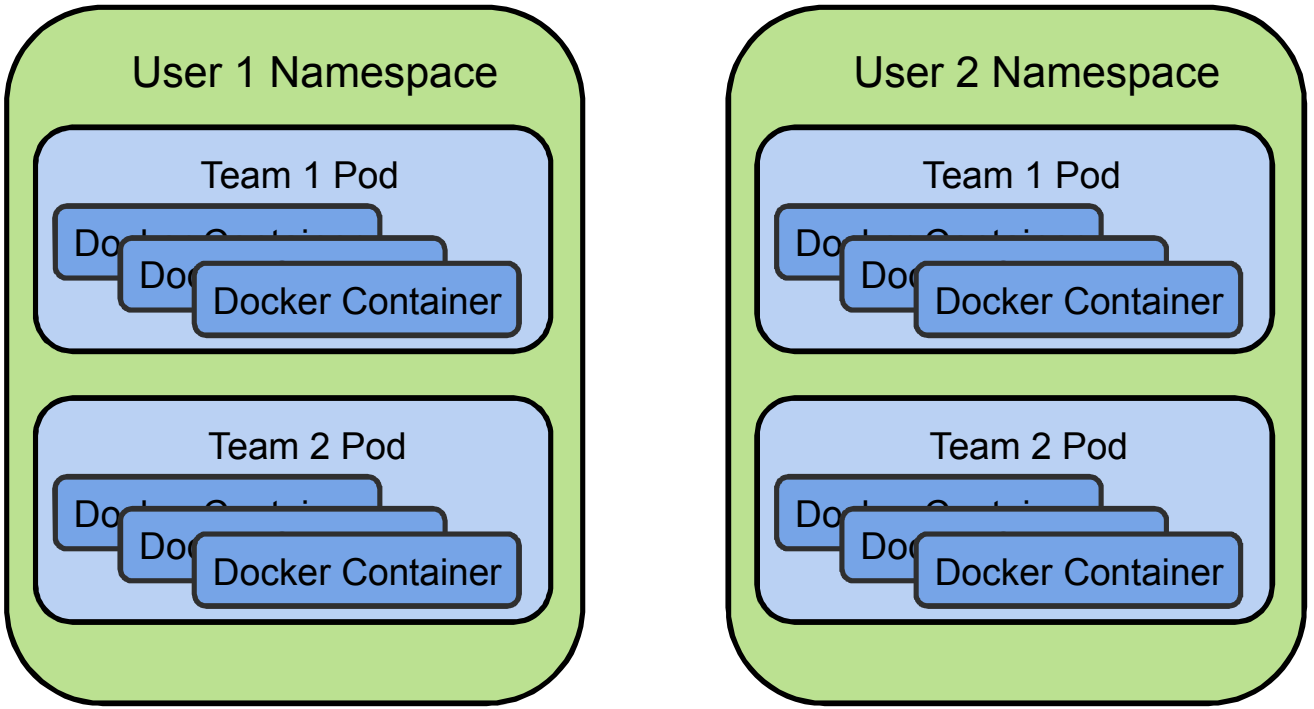
Coordinating Between Teams

Sandia National Laboratories

How to organize multiple groups in a Kubernetes cluster?

- Running multiple instances on the same machines
- Avoiding resource conflicts
- Avoiding configuration conflicts
- Avoiding service discovery conflicts

Coordination Between Teams



- Separate pod file for each team
  - Groups a team's containers on the same machine
  - Team's containers can share resources like shared memory
- Separate Kubernetes namespace for each user
  - Allows multiple instances of same pod without conflict
  - Every developer can run full system on same machine cluster

Coordination Between Teams

Sandia National Laboratories

- Avoiding resource conflicts
  - Resources tied to pods (e.g. Kubernetes emptydir)
    - Each instance runs in separate pod
- Avoiding configuration conflicts
  - Separate root “username” in distributed data-store
- Avoiding service discovery conflicts
  - Separate root “username” in distributed data-store
- Top-level startup sequence establishes root “usernames”

Logging

Sandia National Laboratories

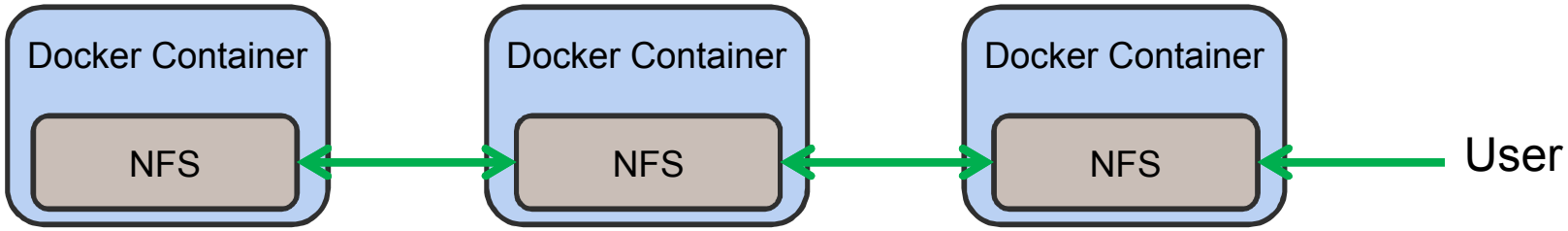
How to store and view logs from distributed applications?

- Retain logs if host VM is deleted
- View logs if you don’t have access to the host machine
- Collect and centralize logs

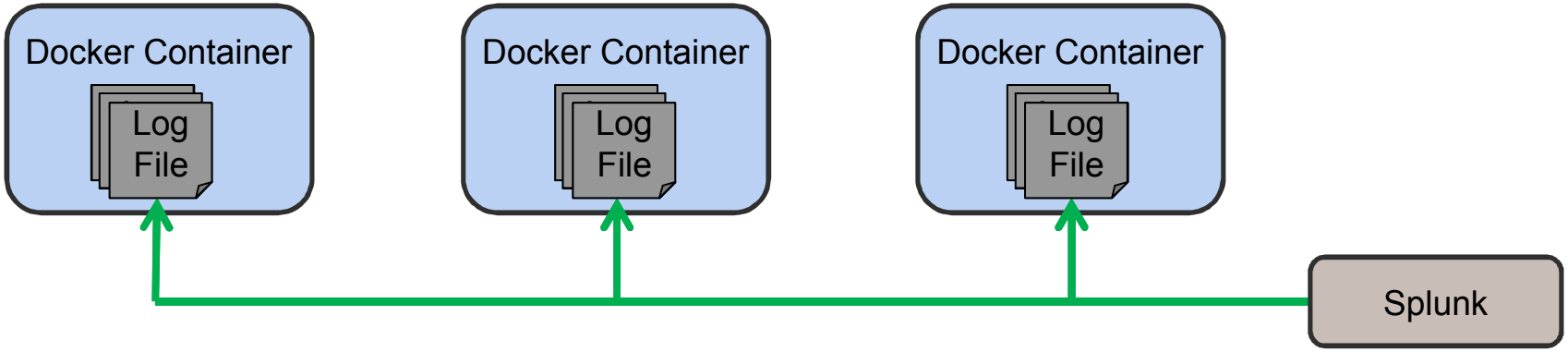
Logging

Sandia National Laboratories

Store logs in NFS directory



Aggregate logs using Splunk



Possible solutions:

- Use NFS to make log files accessible from any VM
- Collect log files into central location
- Collect stdout into central location

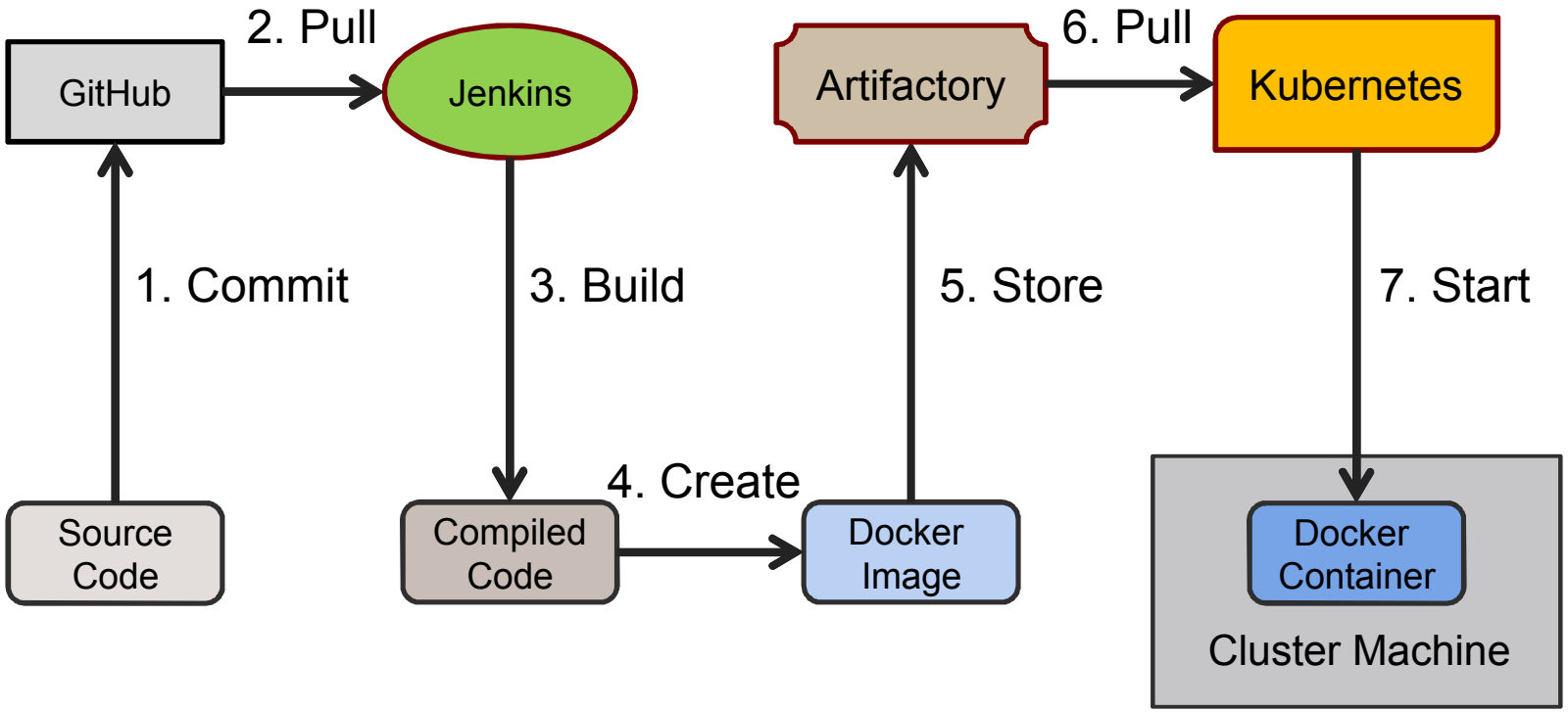
Container Deployment

Sandia National Laboratories

How are container images stored and distributed?

- Who builds the images?
- How does Kubernetes access the images?
- How are the images versioned?

Container Deployment



- Use Jenkins to compile and build images automatically
- Store versioned images in Artifactory
- Pull images to scheduled machines via Kubernetes

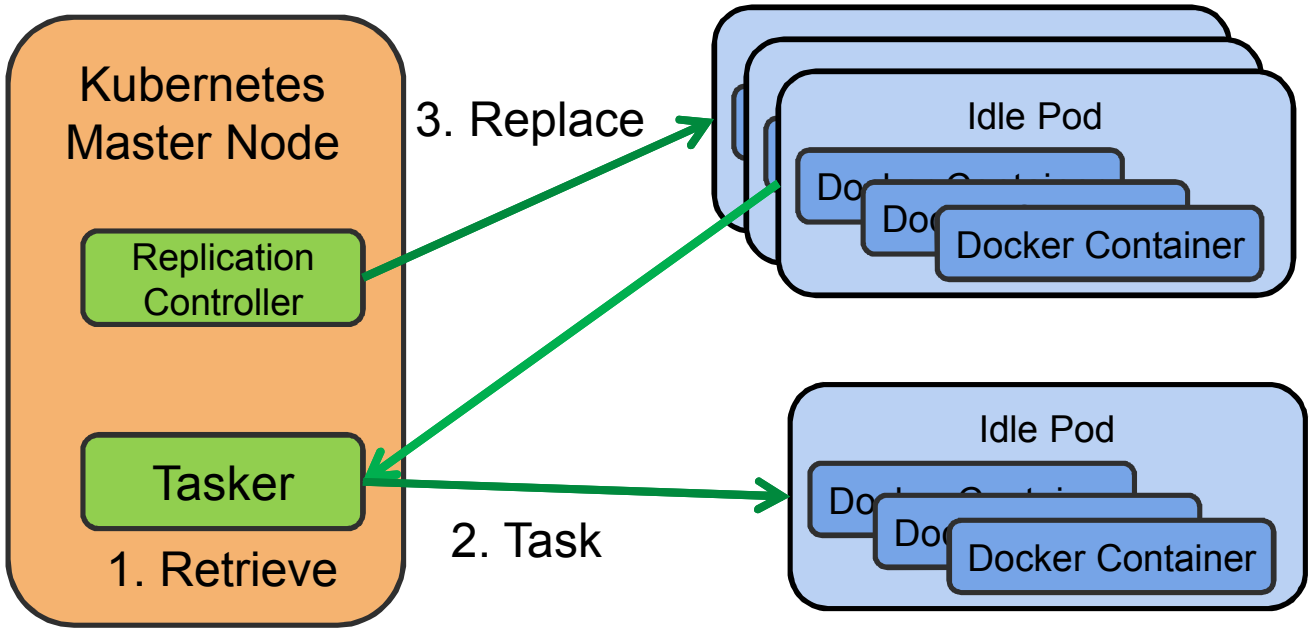
High-Speed Container Startup

Sandia National Laboratories

How to start containers fast enough for dynamic processing?

- Docker startup takes ~100ms
- Kubernetes pod startup takes ~5s

High-Speed Container Startup



One possibility:

- Create pool of idle containers
- Assign a task to a pod
- Kubernetes automatically replaces the pod

Conclusion

- Moving to Docker microservices is complicated
- Standardizing development and deployment is crucial
- High-speed/real-time activities can be tricky
- Workflow is different – there is a learning curve for developers