Global Operations with Docker Enterprise

Nico Kabar

Solutions Architect @ Docker

@nicolakabar

Who's running Docker Enterprise in production?

Who's running Docker Enterprise in production in multiple datacenters?

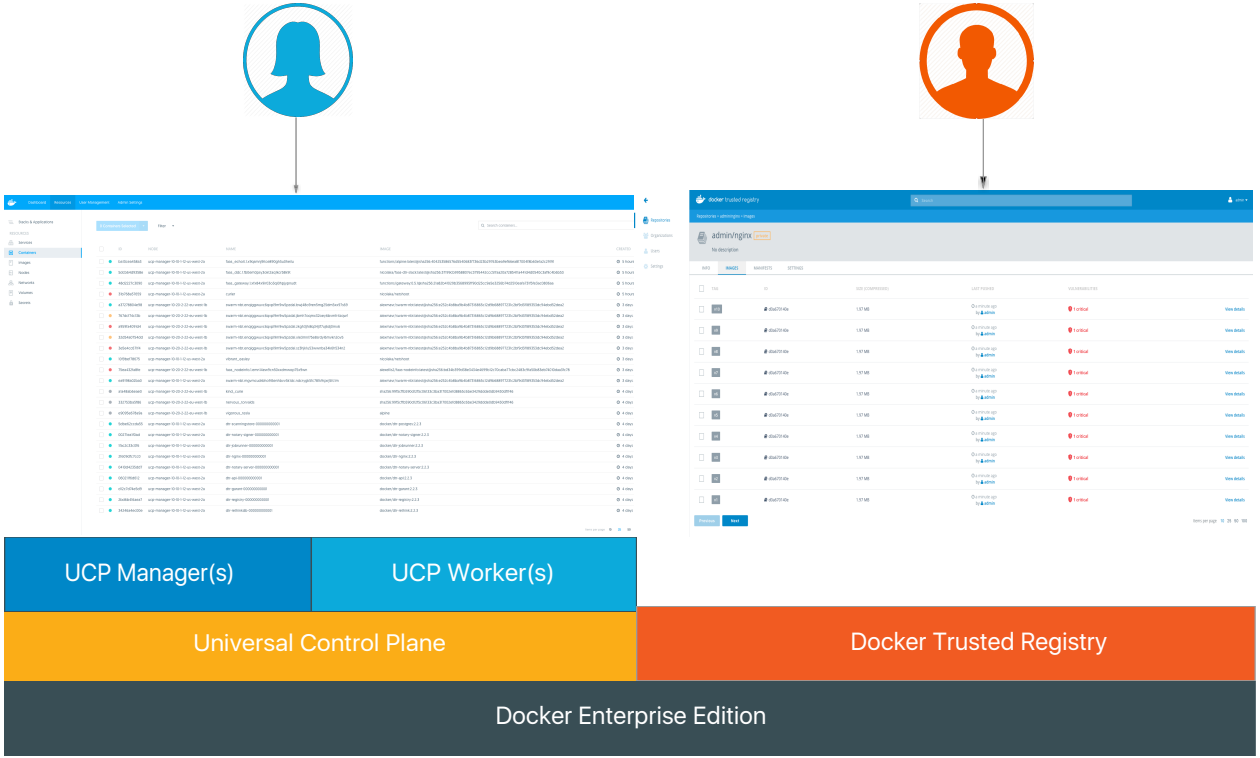
Who's planning on running Docker Enterprise in Production in multiple datacenters?

Agenda

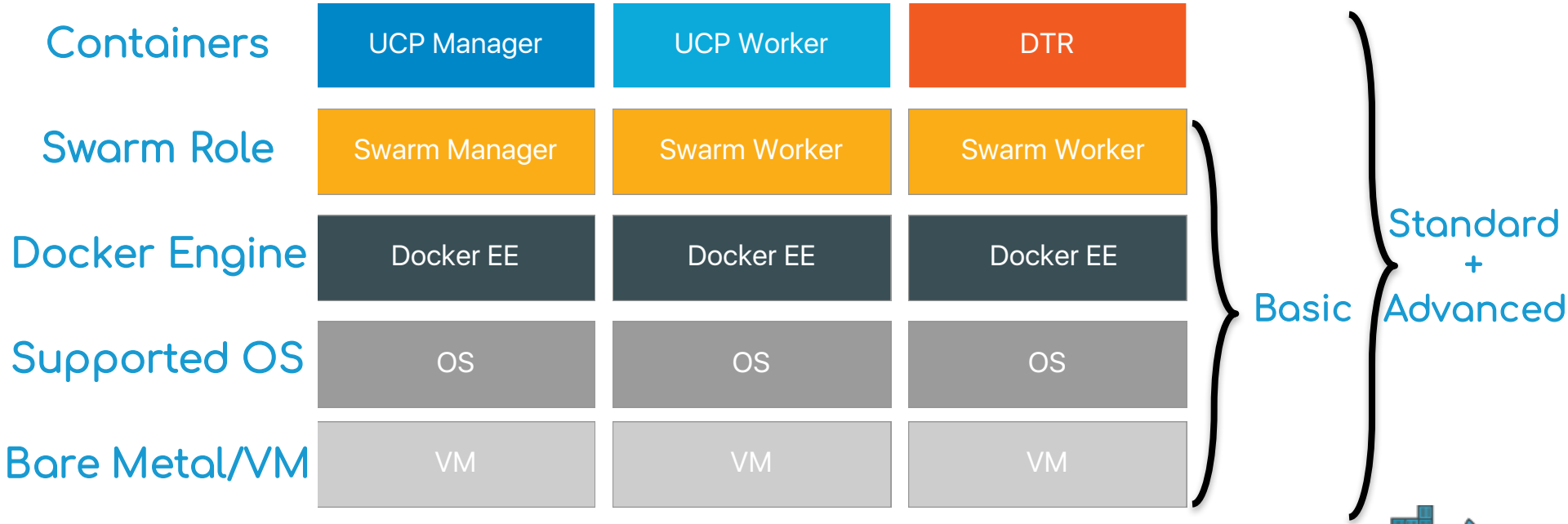
- Docker Enterprise Architecture Overview
- Motivation & Design Consideration
- Building a Global Platform
    1. Robust Infrastructure
    2. Accelerated Image Distribution
    3. Resilient Application Deployment
- Key Takeaways
- Q&A

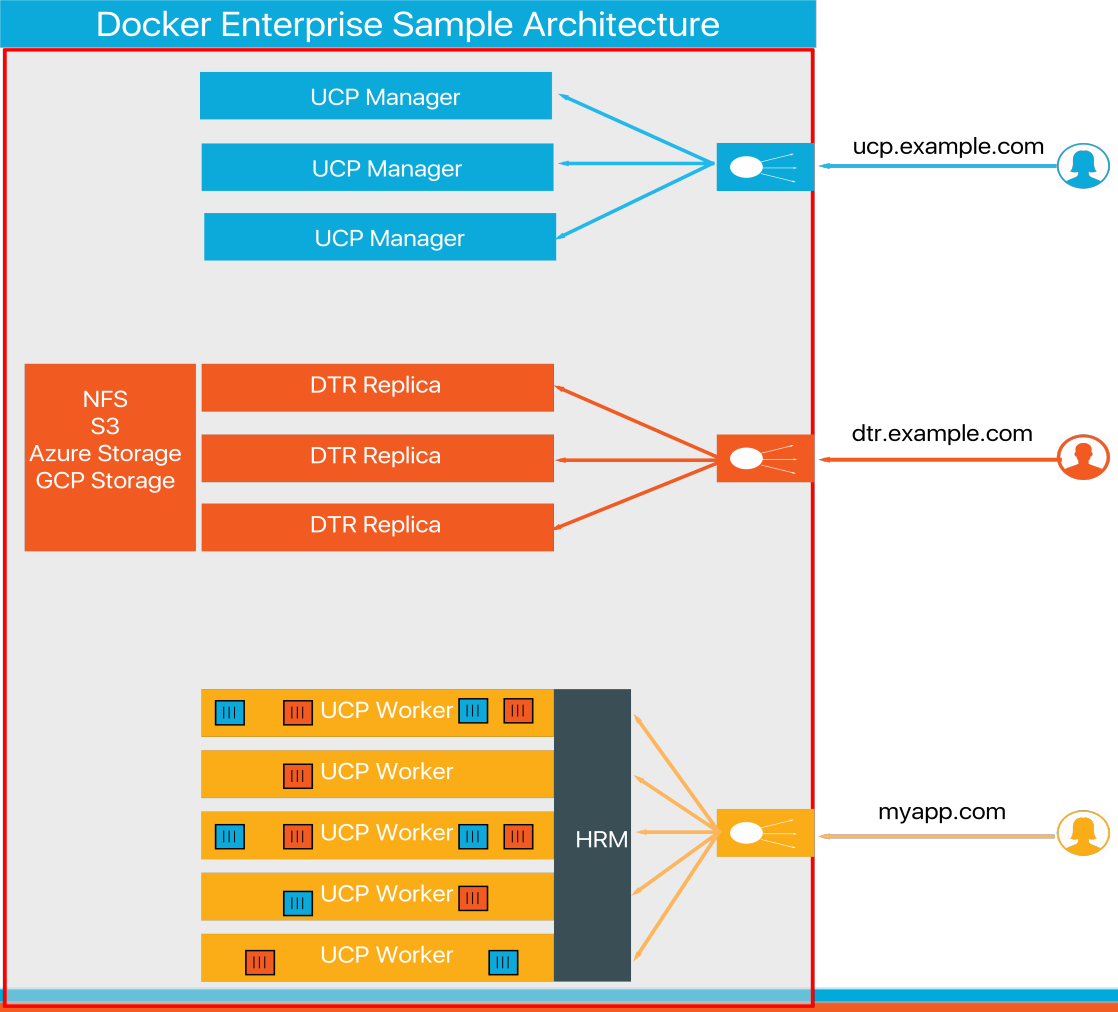
Docker Enterprise Edition Architecture Overview

Docker Enterprise Overview



Docker Enterprise Edition Building Blocks





Docker Enterprise Sample Architecture

Multi-Datacenter Motivation & Key Design Areas

Multi-Datacenter Motivation

- Application Resiliency
- Redundancy
- Disaster Mitigation
- Hybrid Cloud Strategy
- User Experience
- Capacity
- Compliance + Regulations

Let's Build a Global Platform!

Key Design Areas

Robust Infrastructure	Accelerated Image Distribution	Resilient Application Deployments

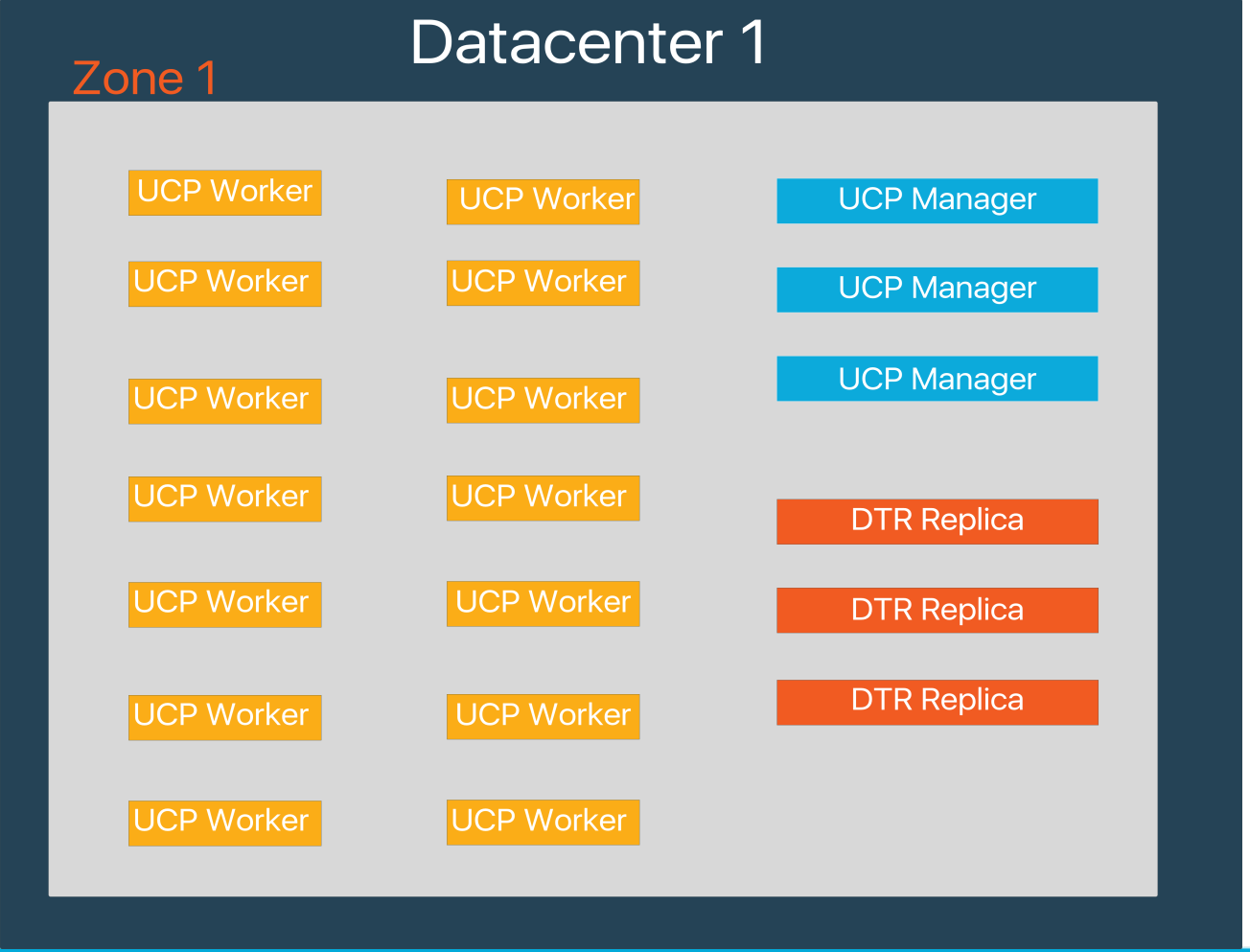
Robust Infrastructure

Number of UCP Managers or DTR Replicas

Number	Majority Requirement	Fault Tolerance
1	1	0
3	2	1
5	3	2
7	4	3

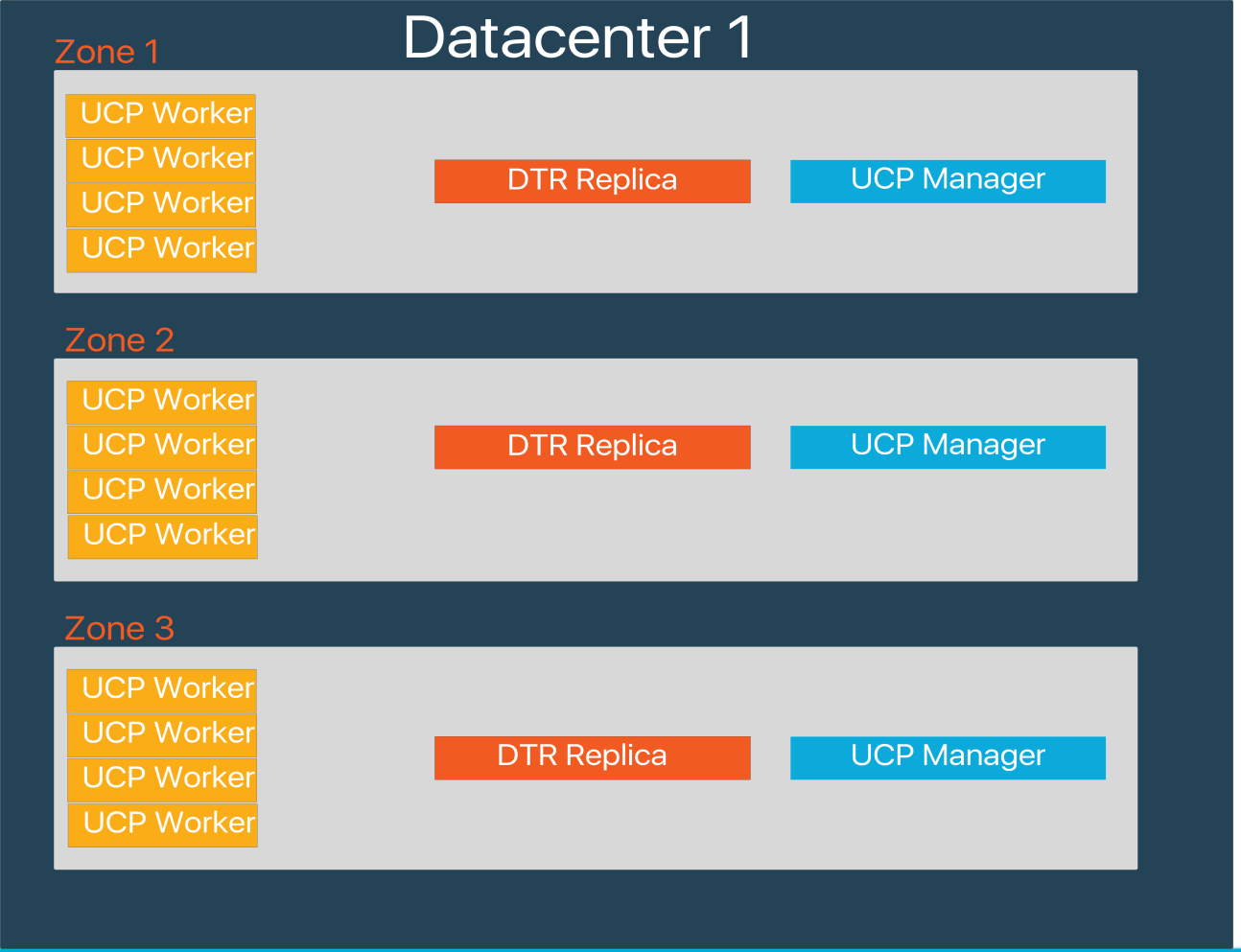
Number of UCP and DTR Clusters

Metric/Cluster Design	Per Stage Only (dev,qa,prod)	Per Datacenter Only (us-west, eu-west)	Per Stage and Datacenter (dev in us-west, dev in eu-west)
Management Complexity	Low	Moderate	High
Resource Utilization	High	High	Moderate
Separation + Isolation	High	Low	High
Network Requirement	Moderate	Low	Low
Infrastructure Requirement	Low	Low	High
Operational Continuity (Docker Content Trust, Image Promotion, RBAC, Application Deployment)	High	Moderate	Low



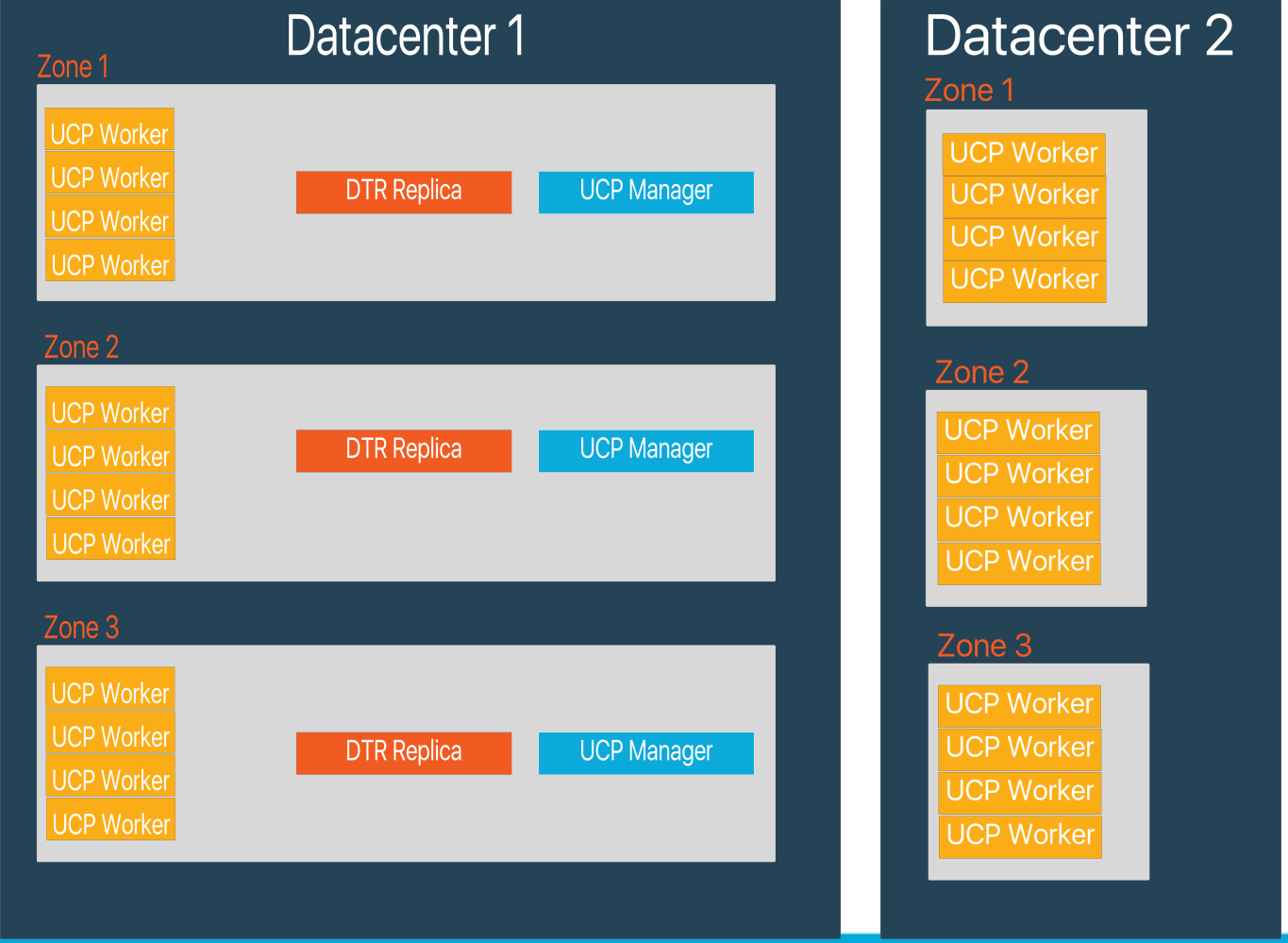
Node Placement

Option 1
Single Zone in a Single Datacenter



Node Placement

Option 2 Multiple Zones in a Single Datacenter



Node Placement

Option 3 Multiple Zones in Multiple-Datacenter

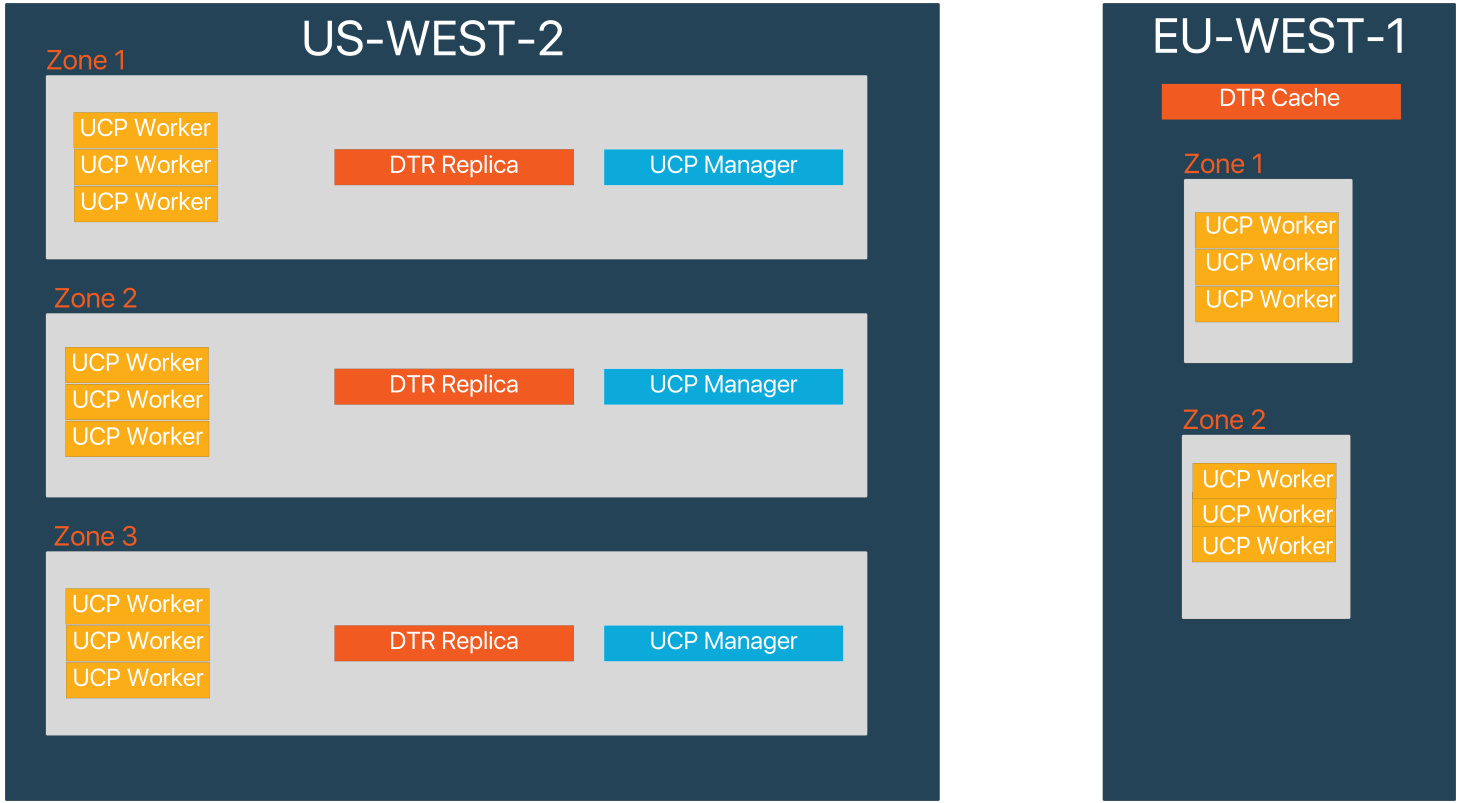
Network Latency Considerations

Role	Component	Heartbeat (ms)	Timeout (ms)	Impact
Swarm Manager	Raft (SwarmKit)*	1000	3000	swarm ops
Swarm Manager, Swarm Worker	Gossip (libnetwork)	1000	5000	overlay, ingress, routing mesh, dns, lb
UCP Manager	Etcd*	500	5000	UCP ops.
UCP Manager, DTR	Rethink*	2000	10000	UCP and DTR Ops.

* Quorum Based

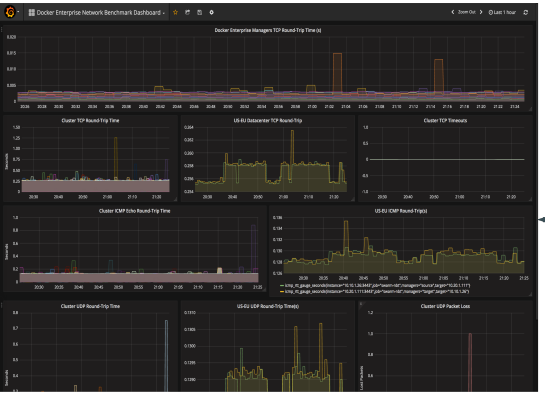
Swarm Network Monitoring

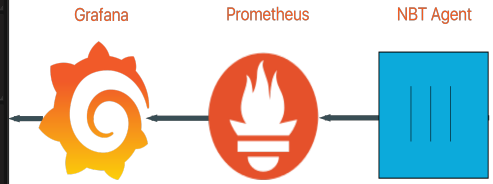
Multi-Datacenter Cluster

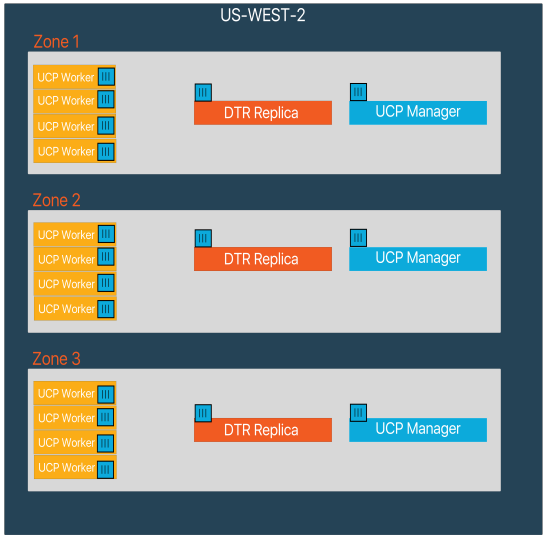


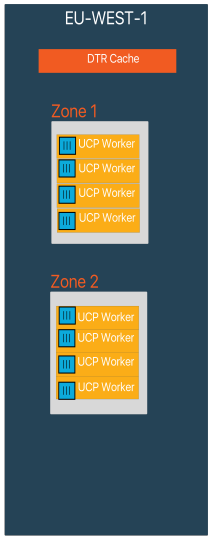
Swarm Network Monitoring

https://github.com/alexmavr/swarm-nbt

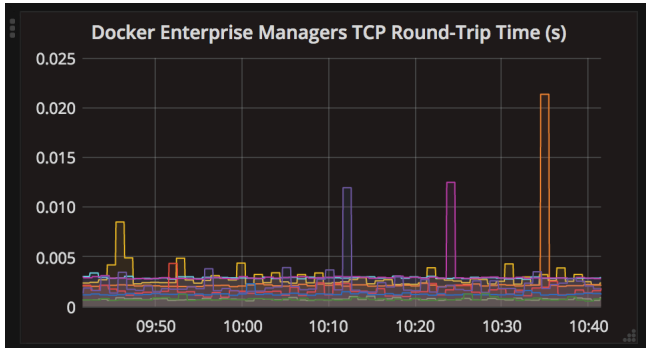


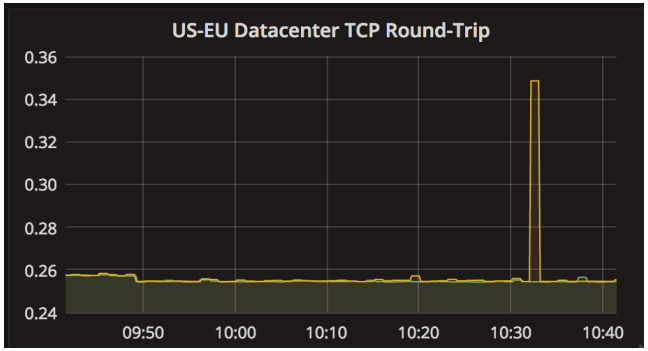


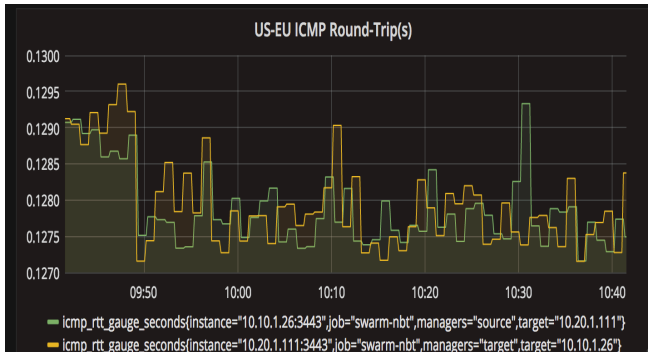


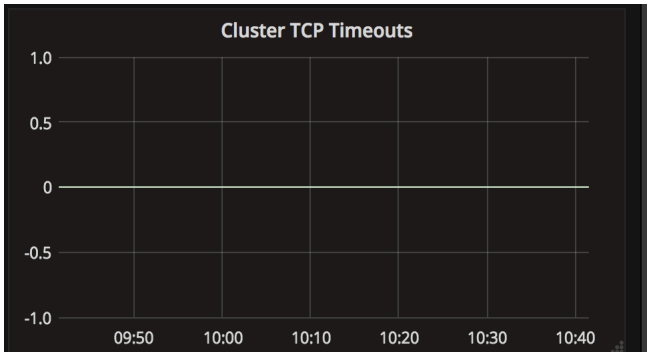


Network Metrics









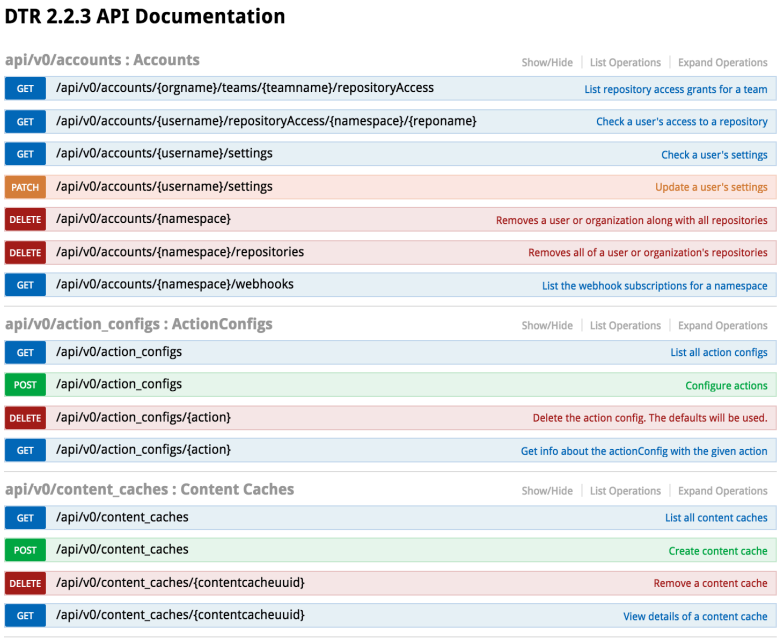
Accelerated Image Distribution

DTR Backend Storage

- Filesystem
  - Local
  - NFS
- Object (Recommended)
  - AWS S3/Compatible
  - Microsoft Azure Blob Storage
  - Google Cloud Storage
  - OpenStack Swift

DTR API

- Configurations(Certs, Storage,Proxies..etc)
- Organization/Account CRUD
- Repository CRUD
- Image Scanning
- Outgoing Webhooks
- Events



DTR 2.2.3 API Documentation

api/v0/accounts : Accounts — Show/Hide | List Operations | Expand Operations

GET	/api/v0/accounts/{orgname}/teams/{teamname}/repositoryAccess	List repository access grants for a team
GET	/api/v0/accounts/{username}/repositoryAccess/{namespace}/{reponame}	Check a user's access to a repository
GET	/api/v0/accounts/{username}/settings	Check a user's settings
PATCH	/api/v0/accounts/{username}/settings	Update a user's settings
DELETE	/api/v0/accounts/{namespace}	Removes a user or organization along with all repositories
DELETE	/api/v0/accounts/{namespace}/repositories	Removes all of a user or organization's repositories
GET	/api/v0/accounts/{namespace}/webhooks	List the webhook subscriptions for a namespace

api/v0/action_configs : ActionConfigs — Show/Hide | List Operations | Expand Operations

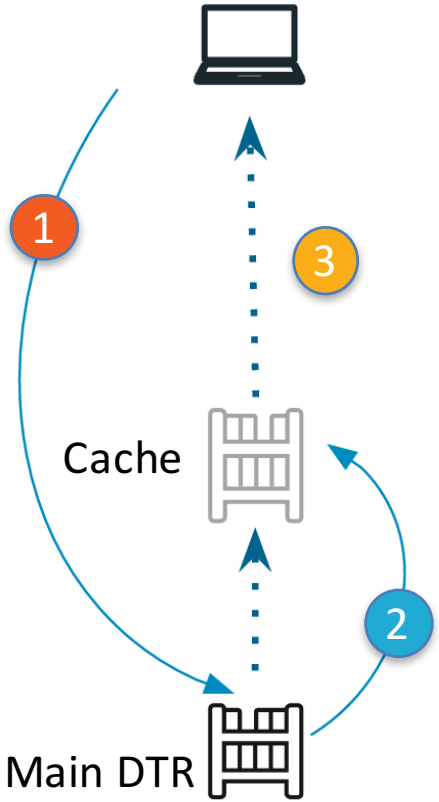
GET	/api/v0/action_configs	List all action configs
POST	/api/v0/action_configs	Configure actions
DELETE	/api/v0/action_configs/{action}	Delete the action config. The defaults will be used.
GET	/api/v0/action_configs/{action}	Get info about the actionConfig with the given action

api/v0/content_caches : Content Caches — Show/Hide | List Operations | Expand Operations

GET	/api/v0/content_caches	List all content caches
POST	/api/v0/content_caches	Create content cache
DELETE	/api/v0/content_caches/{contentcacheuuid}	Remove a content cache
GET	/api/v0/content_caches/{contentcacheuuid}	View details of a content cache

Content Caching

- Faster Pulls
- Reduces WAN Traffic
- Per User Configuration
- Chaining OK
- Configurable TTL
- Supports Same Storage Backbends as DTR
- Advanced Caching (e.g Secure Image Caching)



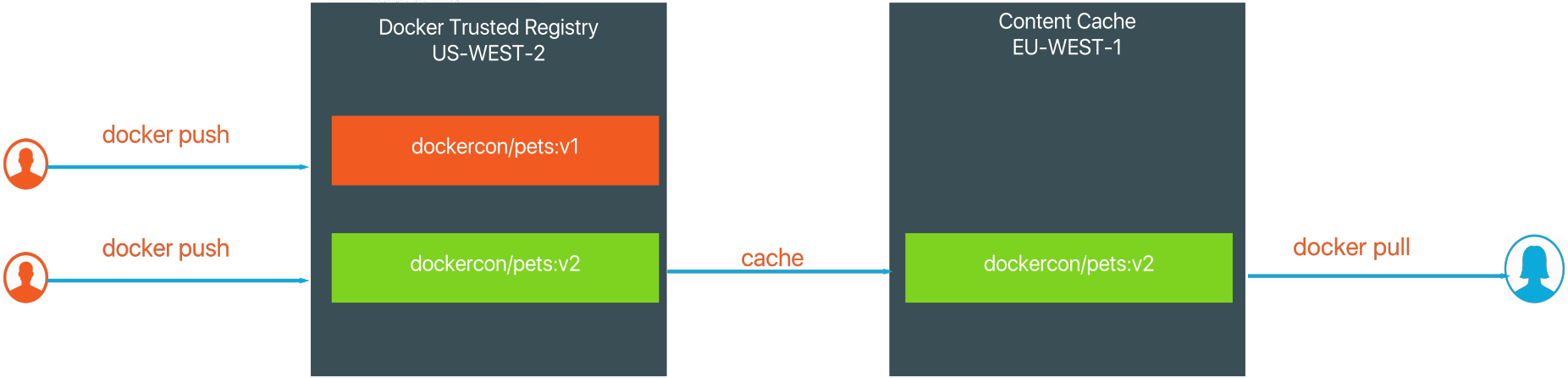
docker17 con

Demo 1
Automating Secure Image Caching

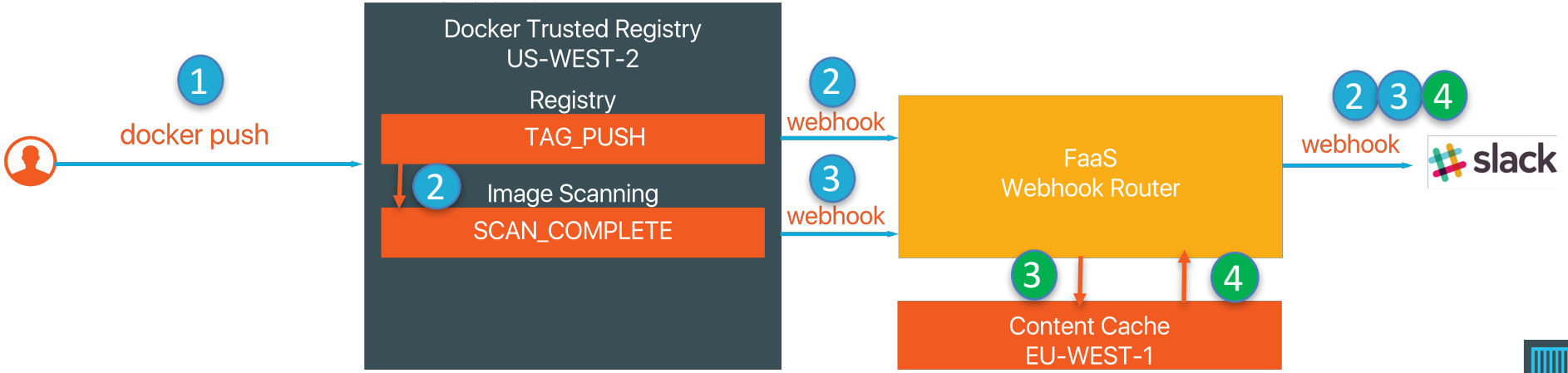
Demo Objectives

1. Integrate DTR Webhooks with Slack
2. Automate DTR Caching based on Security Scanning Results
3. Show Content Caching Results

Demo Details



Demo Details



- Secure Images Only

Resilient Application Deployment

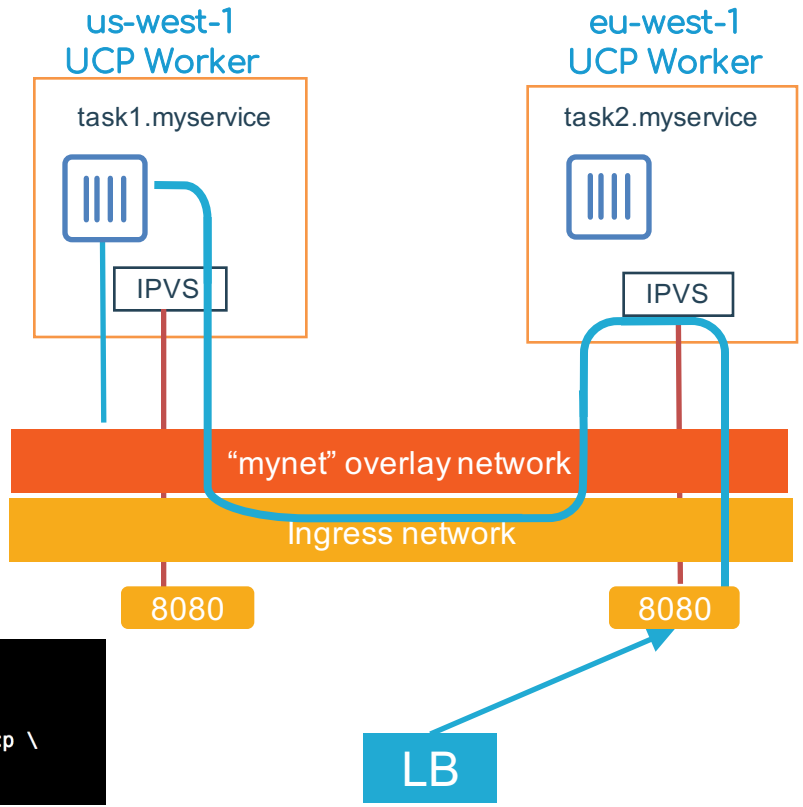
Topology Aware Scheduling

- Placement preference based on node/engine label
- Equal distribution of tasks based on node label
- Hierarchal preference (region > zone > rack)
- Best Effort
- In 17.04-ce & 17.06-ee

```
docker service create \
 --name myapp \
 --replicas 10 \
 --placement-pref 'spread=node.labels.region' \
 --placement-pref 'spread=node.labels.zone' \
 --placement-pref 'spread=node.labels.rack' \
 myapp:v1
```

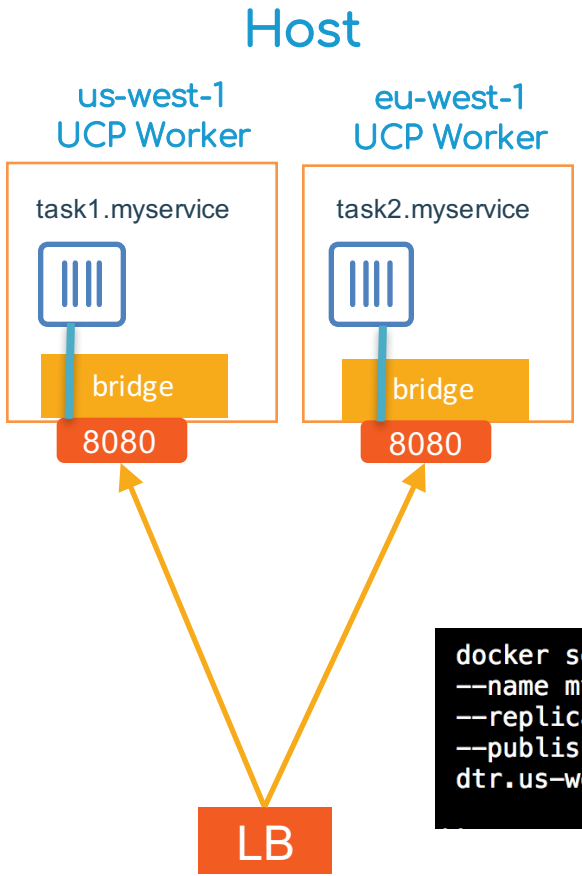
Ingress Networking Mode

- Uses Overlay + IPVS
- Good for East-West traffic
- Potentially bad if you don't control the underlay (public cloud, internet)




```
docker service create \
 --name myapp \
 --replicas 2 \
 --publish mode=ingress,target=5000,published=32000,protocol=tcp \
 dtr.us-west.dcus17.dckr.org/dockercon/myapp:v1
```

Host Networking Mode



- Uses default bridge + NAT to expose service port
- No cross-datacenter networking traffic
- Good for North<>South Traffic

```
docker service create \
--name myapp \
--replicas 2 \
--publish mode=host,target=5000,published=32000,protocol=tcp \
dtr.us-west.dcus17.dckr.org/dockercon/myapp:v1
```

docker17 con

Demo 3

Geo-Aware Application Deployment + Access

Demo Objectives

1. Launch a service in two datacenter
2. Use topology aware scheduling to ensure equal distribution across 2 regions and 5 zones
3. Access the application based geo-DNS

Geo-Aware Service Scheduling

US User EU User

Key Takeaways

- Choose the right architecture that fits your requirements
- Per app-stage clusters recommended
- UCP Managers/DTR Replicas in same region
- UCP Workers can extend across regions
- (Keep) Monitoring Your Platform
- Utilize DTR API and webhooks for automation
- Use Content Cache to minimize pull time
- Use topology aware scheduling for enhanced scheduling
- Use the networking model that best suits your applications

The Docker Success Center provides expert troubleshooting and advice for Docker EE customers. From troubleshooting to best practices and security considerations, we've got you covered.

Knowledge Base

Constantly updated with information and guidance from Docker's troubleshooting experts. Get the practical information you need to successfully manage and troubleshoot your Docker environment.

Learn More

Reference Architectures

Expert advice and guidance from Docker's top practitioners. Get design considerations, best practices, and decision support for architecting and building your Docker environment.

Learn More

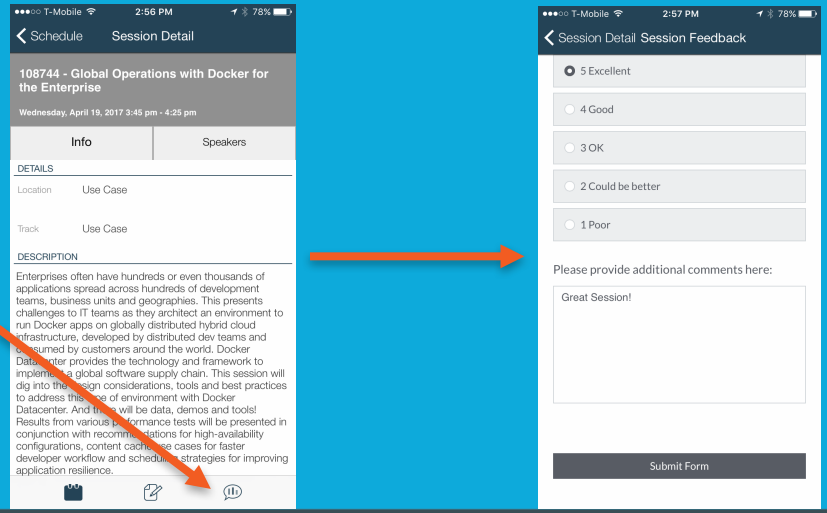
Technical Support

Access Docker's SLA-based enterprise technical support. Expert assistance you can rely on for enterprise deployments.

Learn More

Thanks! Questions?

Feedback Welcome!



@docker #dockercon

docker17 con