INTRO TO DOCKER FOR ENTERPRISE OPERATIONS

DOCKERCON 2017 - AUSTIN

A NOTE ON PEDAGOGY

- Docker believes in learning by doing, with support.
- Work with your colleagues to solve problems, and don't hesitate to interrupt the lecture for clarification.

SESSION LOGISTICS

- 3 hours duration
- mostly exercises

ASSUMED KNOWLEDGE AND REQUIREMENTS

- Familiarity with using the Linux command line
- A UCP License (download one at https://store.docker.com/bundles/docker-datacenter/purchase?plan=free-trial)
- You should know the basics of Docker
  - Run a Docker container
  - Search for and pull images from Docker Store
  - Use Docker for Mac / Windows on your local machine

YOUR LAB ENVIRONMENT

- You have been given several instances for use in exercises.
- Ask instructor for access credentials if you don't have them already.

AGENDA

- Universal Control Plane
  - Application management
  - User management
  - Datacenter architecture
- Docker Trusted Registry
  - Image sharing
  - Image scanning
- ...plus other odds and ends.

INTRODUCTION TO DOCKER DATACENTER

THE SOFTWARE SUPPLY CHAIN

- Image Creation
- Image Distribution
- Container Execution

Docker Datacenter enables security, ease of use, and enterprise-grade control at each of these steps.

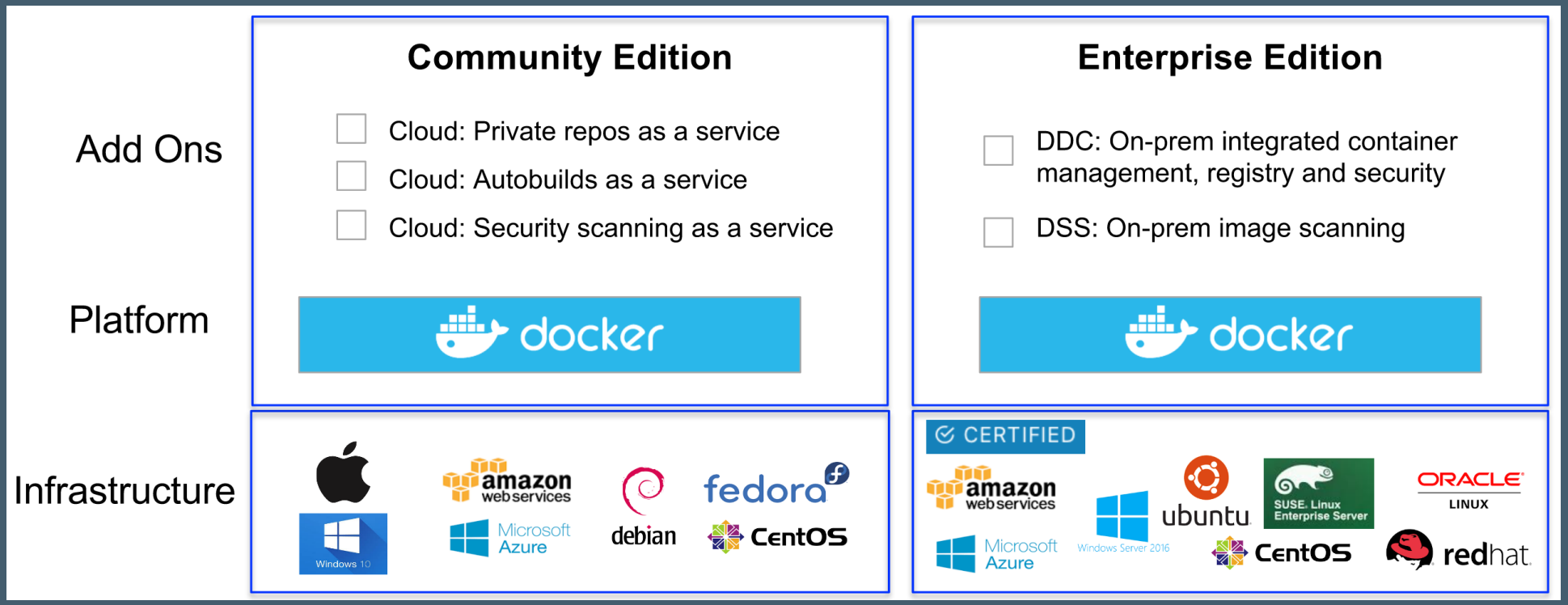
WHAT SHOULD OPERATIONS KNOW ABOUT DOCKER?

Containers are:

- Lightweight (therefore dense)
- Ephemeral & (ideally) stateless (therefore scalable as Services, monitored in aggregate)
- Portable (therefore distributable over Swarms)
- Immutable in production

DOCKER DATACENTER

Integrated, end-to-end platform for agile application development and management in production



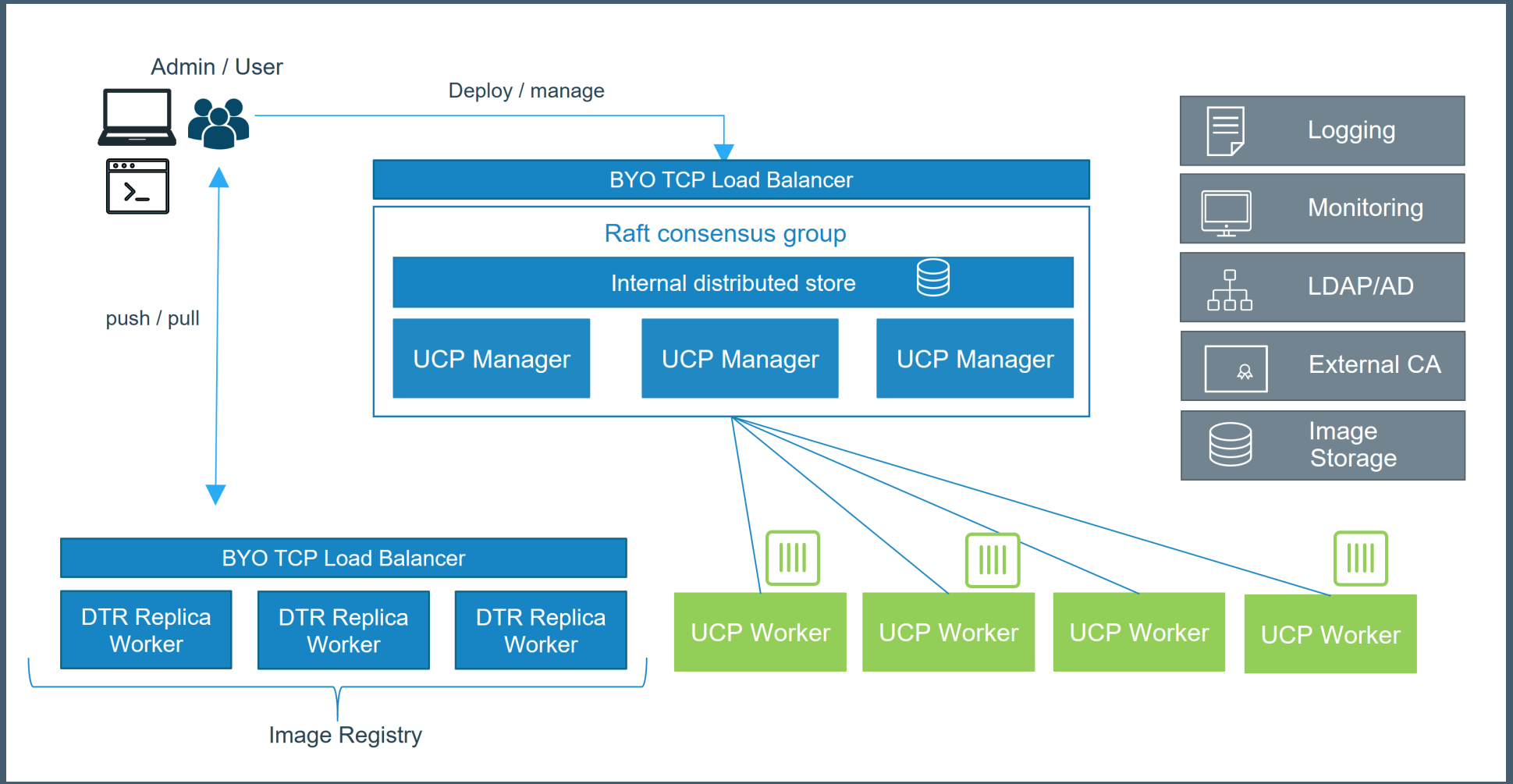
INTEGRATION: BATTERIES INCLUDED BUT SWAPPABLE

- Certificate authorities
- Network drivers
- Storage backends
- User management
- Monitoring
- ...

KEY DDC FEATURES

- Build:
  - Security Scanning
  - Webhooks
- Ship:
  - Content Trust
  - Content Cache
- Run:
  - Role based access control
  - Universal Control Plane (GUI)

DOCKER DATACENTER ARCHITECTURE



EXERCISE: CONFIGURE UCP

Work through the 'Configure UCP' exercise in the Docker for Enterprise Operations Exercises book.

UNIVERSAL CONTROL PLANE

DOCKER SWARM MODE REVIEW

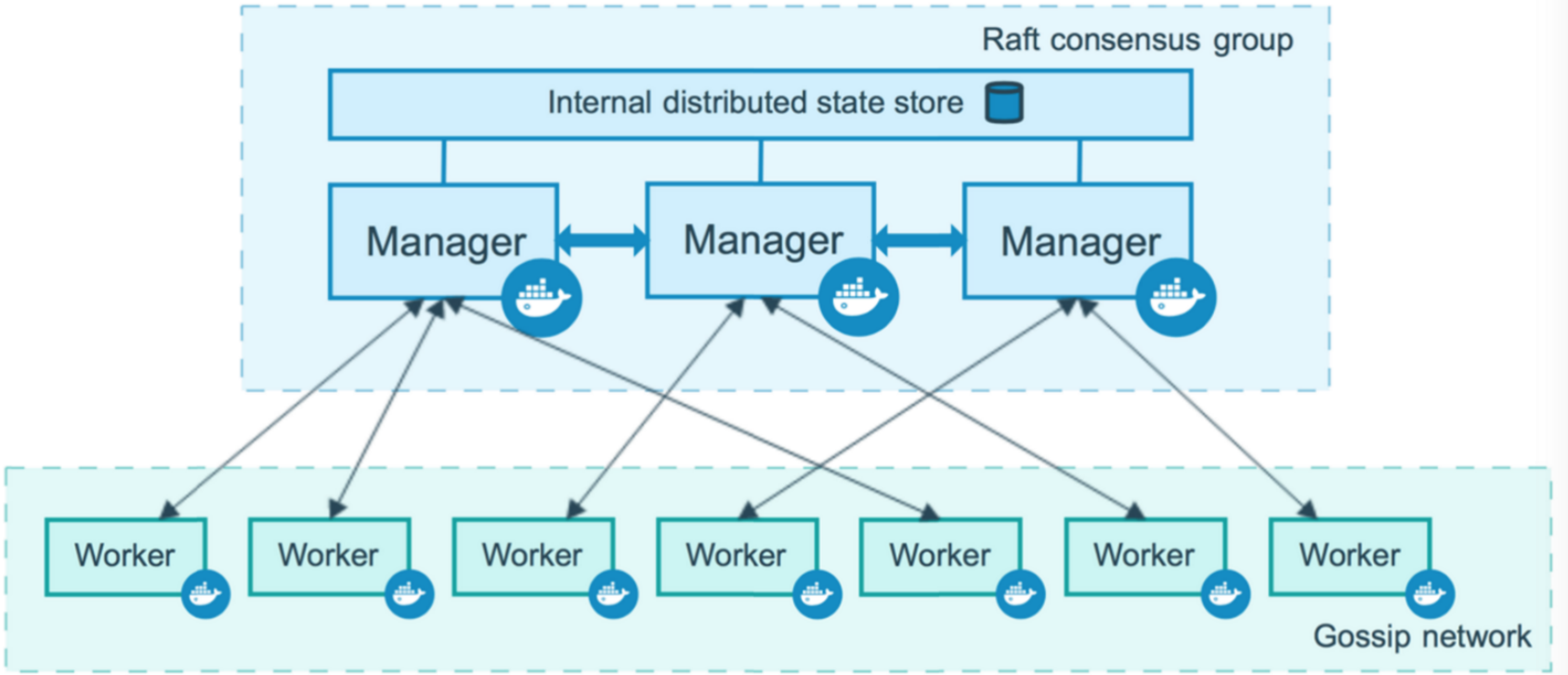
- Based on Swarmkit
- Native Docker orchestration
- UCP is built on top of Swarm

SWARM CONCEPTS

SWARMS, NODES, SERVICES & TASKS

- **Swarm**: a collection of Docker engines accepting `docker container run` commands from a scheduler.
- **Node**: instance of engine; either manager or worker.
- **Service**: the operational definition of an app (image + config).
- **Task**: an instance of our app (container)

SWARM MODE ARCHITECTURE



MULTIPLE MANAGER NODES

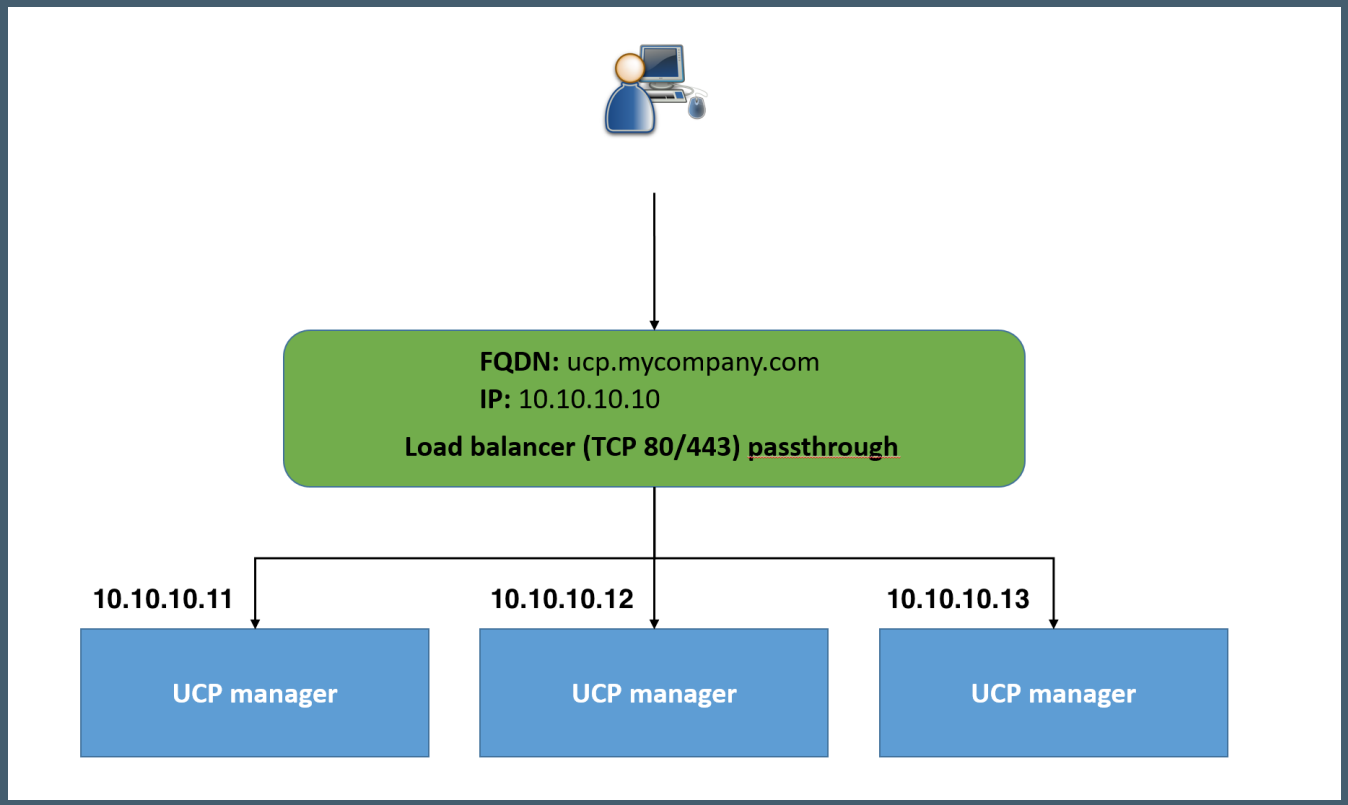
- High availability / fault tolerance
- Performance tradeoff
- Always odd number
- Load balancing

HOW MANY MANAGERS ?

Number of manager nodes	Failures tolerated	Situation
1	0	Dev environment
3	1	Small deployment
5	2	Small-Medium
7	3	Medium-Large
N	(N-1) / 2	

- **Note:** Having a 3 manager node setup does not provide true failover

LOAD BALANCING EXAMPLE



- Balance ports 443 + any exposed services
- Don't terminate HTTPS
- Use `/_ping` endpoint for health check

UCP IS A SWARM

- UCP sits on top of a Swarm
- Adds UI, RBAC, registry integration...
- Start with one UCP manager == Swarm manager leader
- New UCP nodes joined exactly as new Swarm nodes joined

EXERCISE: UCP HIGH AVAILABILITY

Work through the 'Adding UCP Manager Nodes' exercise in the Docker for Enterprise Operations Exercises book.

UCP APPLICATION COMPONENTS

- Extensions to Swarm provided by a collection of containers
- Core component: global service `ucp-agent`
- `ucp-agent` deploys and keep-alive all other UCP containers

MANAGER NODE CONTAINERS

- **`ucp-controller`** - The UCP web server
- **`ucp-auth-api`** - The centralized service for identity and authentication used by UCP and DTR
- **`ucp-auth-store`** - Stores authentication configurations, and data for users, organizations and teams
- **`ucp-auth-worker`** - Performs scheduled LDAP synchronizations and cleans authentication and authorization data
- **`ucp-client-root-ca`** - A certificate authority to sign client bundles
- **`ucp-cluster-root-ca`** - A certificate authority used for TLS communication between UCP components
- **`ucp-kv`** - etcd service used to store the UCP configurations
- **`ucp-proxy`** - A TLS proxy. It allows secure access to the local Docker engine to UCP components
- **`ucp-swarm-manager`** - Manager container for Docker Swarm (classic). Provides backwards compatability

UCP VOLUMES

Named volumes used to persist data:

- **`ucp-auth-api-certs`** - Certificate and keys for the authentication and authorization service
- **`ucp-auth-store-certs`** - Certificate and keys for the authentication and authorization store
- **`ucp-auth-store-data`** - Data of the authentication and authorization store
- **`ucp-auth-worker-certs`** - Certificate and keys for authentication worker
- **`ucp-auth-worker-data`** - Data of the authentication worker
- **`ucp-client-root-ca`** - Root key material for the UCP root CA that issues client certificates
- **`ucp-cluster-root-ca`** - Root key material for the UCP root CA that issues certificates for swarm members

UCP VOLUMES (CONT'D)

- **`ucp-controller-client-certs`** - Certificate and keys used by the UCP web server to communicate with other UCP components
- **`ucp-controller-server-certs`** - Certificate and keys for the UCP web server running in the node
- **`ucp-kv`** - UCP configuration data
- **`ucp-kv-certs`** - Certificates and keys for the key-value store
- **`ucp-node-certs`** - Certificate and keys for node communication

UCP USER MANAGEMENT AND ACCESS CONTROL

USER MANAGEMENT OVERVIEW

Permissions are conferred in two steps:

1. User Default Permissions: read/write privileges for a given user
2. Team Permissions: extend defaults for groups of users on subsets of objects

USER DEFAULT PERMISSION LEVELS

- Admin: no restrictions
- Non admin: permissions depend on **Default Permissions**:
  - No Access: no read or create permission
  - View Only: view volumes, networks, images, secrets
  - Restricted Control: R/W volumes, networks, images, secrets; create unprivileged containers
  - Full Control: Restricted + `exec`, privileged containers

Note only admins can see containers created by other users.

TEAM PERMISSIONS

- UCP users can be assigned to **teams**
- All objects in UCP can bear **labels**
- Team membership confers special permissions on a label-by-label basis

Example:

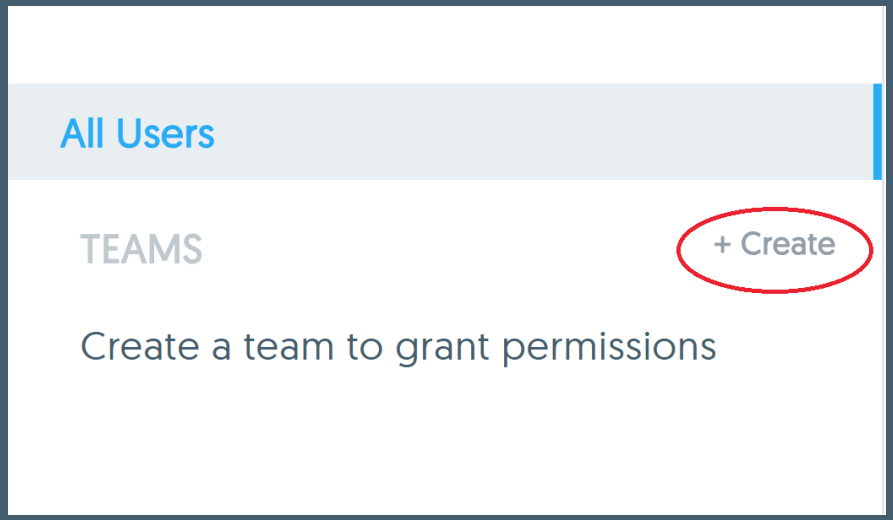
- Lisa is a member of the engineering team.
- The engineering team has Restricted Control access to the dev label.
- Therefore, Lisa has *at least* Restricted Control privilege to anything with the dev label.

TEAM PERMISSIONS

- No Access: team members have no access permissions
- View Only: team members can view objects with this label
- Restricted Control: team members can view + create containers with this label (no privileged containers, no exec)
- Full Control: team members can create containers with this label with no restriction

CREATING USERS & TEAMS

- Created in UCP or synced from LDAP
- Users can belong to multiple teams



LDAP INTEGRATION

- Delegate authentication to an LDAP server
- User accounts will be synced from LDAP based on a LDAP search configuration
- Existing accounts setup in UCP will be disabled except for one recovery admin account
- Recovery admin account allows for UCP admin access in case of a problem with the LDAP server
- Once enabled, users cannot change their password in UCP

DEPLOYING SERVICES WITH LABELS

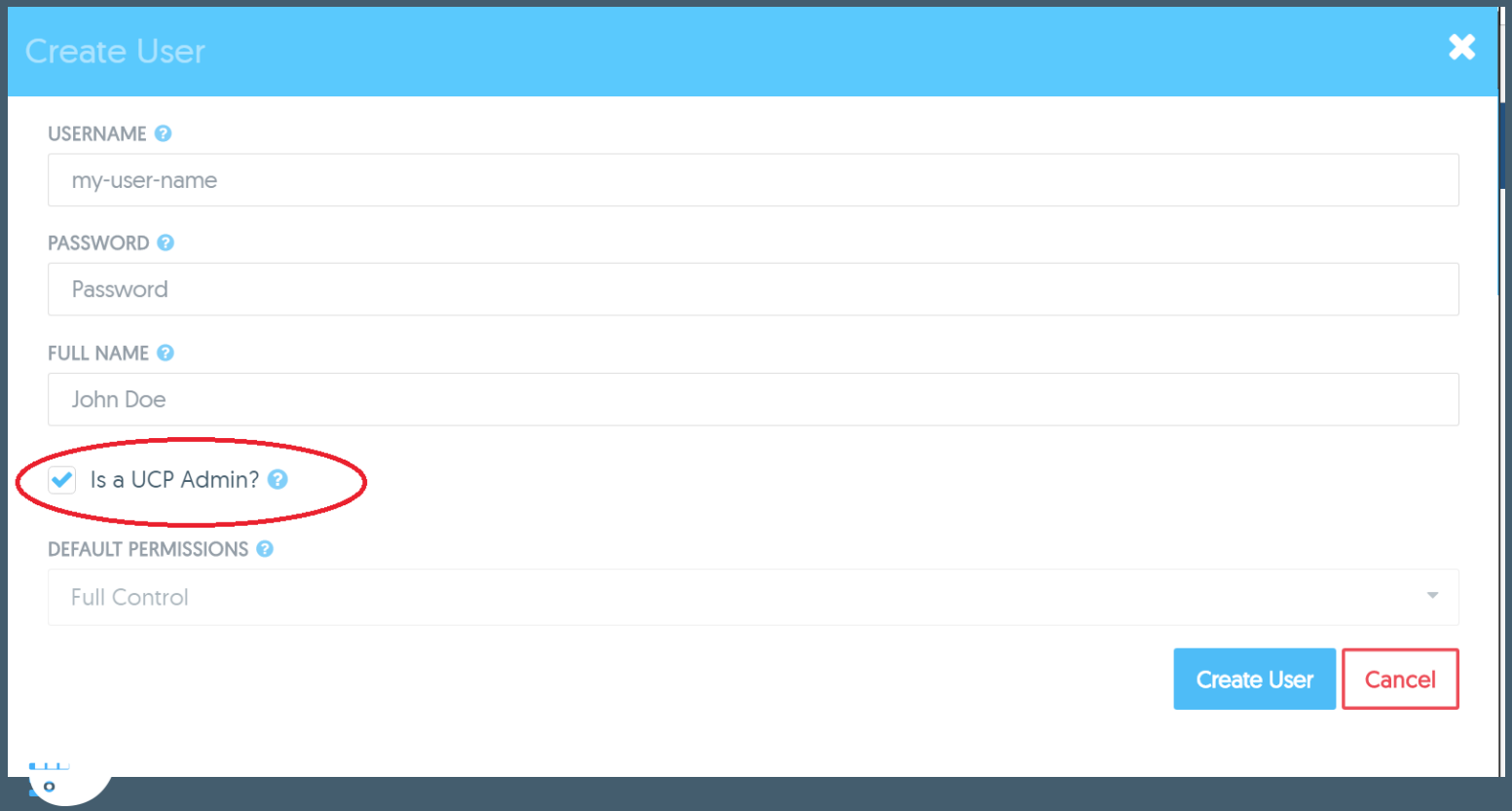
- Containers carry the same labels as their parent service
- Therefore, user must apply a label with at least Restricted Control to a service on creation
- Unlabeled containers can only be deployed if user's default permission is at least Restricted Control

RESTRICTED CONTROL

- Similar to Full control but with a few key restrictions
- Allows users to create, restart, kill or remove containers
- Does not give permission to `docker container exec` into a container
- Also prevents running:
  - Privileged containers
  - Host mounted volumes

ADMIN USERS

- Are authorized to access all Docker objects in UCP
- Access either through GUI or client bundle
- Note this is different (higher) than Full Control!



EXERCISE: USER MANAGEMENT

Work through the 'Access Control with Users, Teams and Labels' exercise in the Docker for Enterprise Operations exercise book.

DOCKER TRUSTED REGISTRY

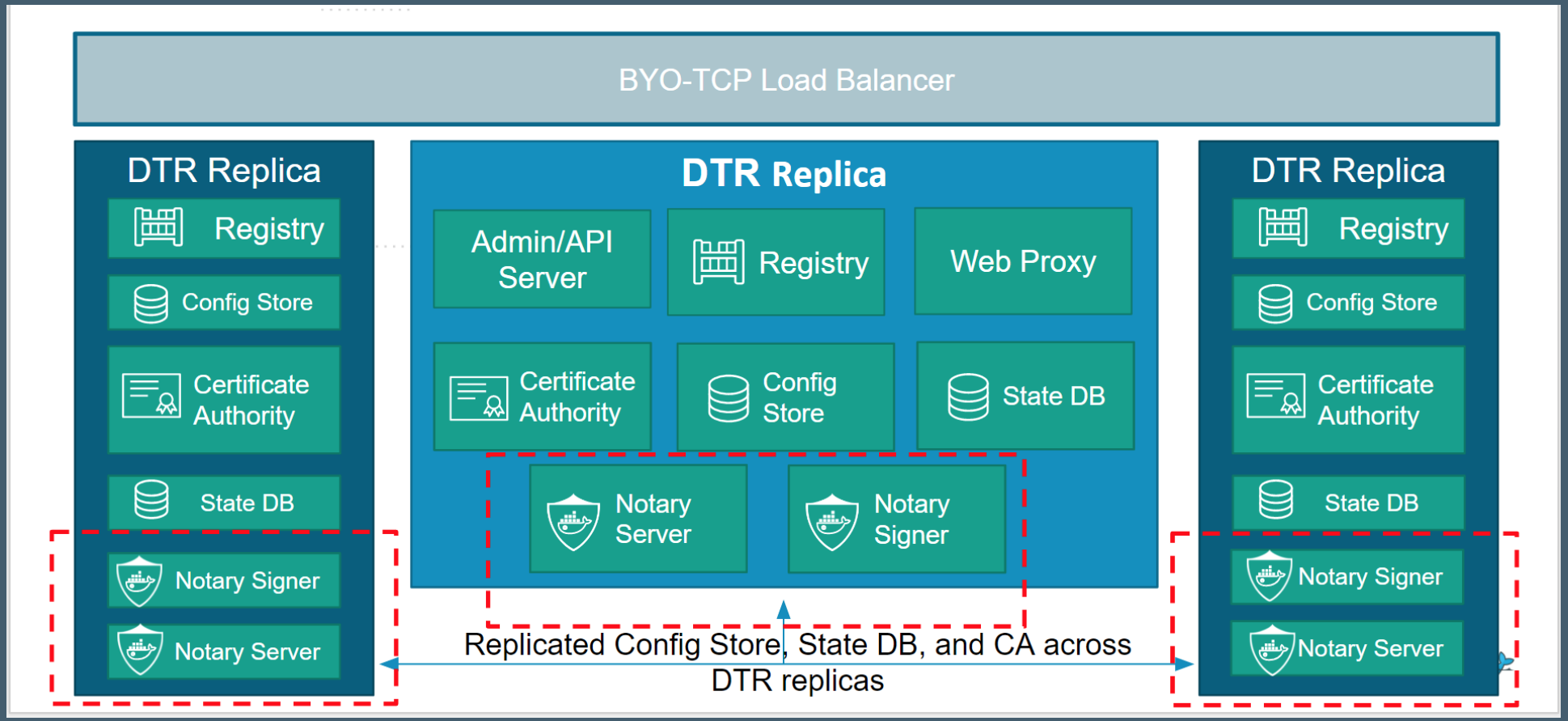
THE SOFTWARE SUPPLY CHAIN

- Image Creation
- Image Distribution
- Container Execution

DTR KEY FEATURES

- Image Creation:
  - Image Security Scanning
  - Webhooks
- Image Distribution:
  - Content Trust / Notary
  - Content Cache
- Image Storage:
  - Pluggable Storage Drivers

DTR ARCHITECTURE



DTR USAGE EXAMPLES

- Containerized CI/CD
  - Registry for images as they move through pipeline
  - Facilitated by DTR Webhooks
  - Enhanced by Image Scanning
  - Developers only push code
- Containers as a Service
  - Curated app catalogue for devs
  - 'Instant-on' environment and services across infra.

INSTRUCTOR DEMO: DTR INSTALLATION

Instructor to demo the process of installing DTR with replicas; see 'Install DTR' and 'Install DTR Replicas' in the exercise book.

EXERCISE: CREATE A PUBLIC REPO

Complete the exercise "Create a DTR Repository" from the Docker for Enterprise Operations exercise handbook

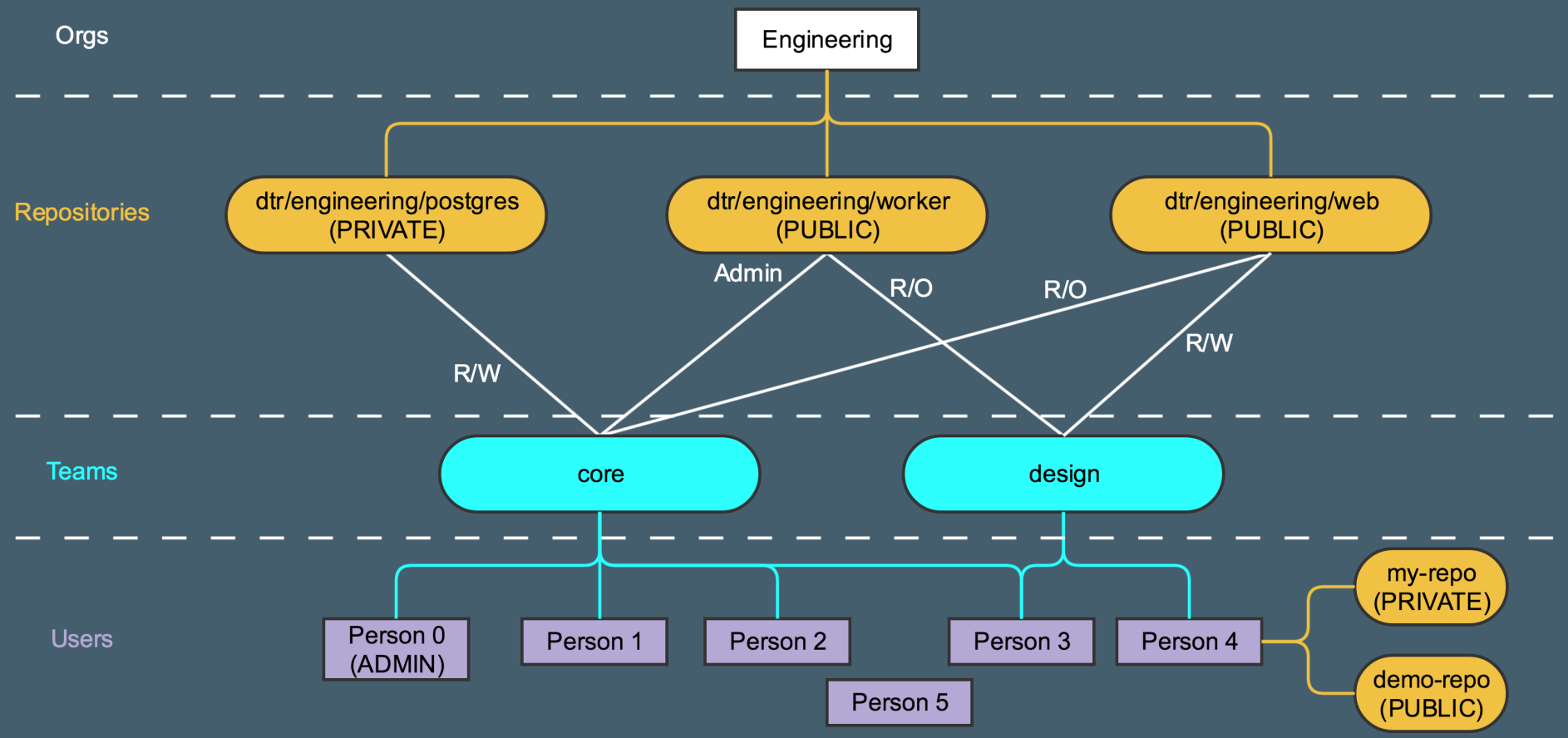
DTR ORGANIZATIONS AND TEAMS

DTR ORG CHARTS

Four key entities:

- **Organizations** namespace all other assets.
- **Repositories** hold images.
- **Teams** define access control to repositories.
- **Users** are grouped by teams and orgs.

A 'SIMPLE' CASE



REPOSITROY PERMISSIONS

Repo access is controlled by two concerns:

- Public vs. Private, and Org vs User owned
- Team Permissions

PUBLIC/PRIVATE/OWNERSHIP MATRIX

	Public	Private
User-Owned	• Pull w/ auth • Visible to all • Push by owner	• Only visible to owner & admins
Org-Owned	• Anyone can pull • Visible to all • Push by R/W team	• Must be R/W or R/O team to see repo

REPOSITORY PERMISSIONS

- Read only
  - Can browse repository and pull images
- Read write
  - Can do everything from the read only permission
  - Push images
  - Delete tags
- Admin
  - Can do everything from read only and read write permissions
  - Edit repository description
  - Set public or private
  - Change team access level

ORGANIZATION MEMBERS

- Users can belong to an organization without belonging to a team
- Team members are automatically organization members
- Organization members can:
  - View other members
  - View organization teams and their members
  - View and pull images from public repositories in the organization
- Organization members do not have the ability to push images to a repository. They must belong to a team with the right permission

ORGANIZATION ADMINS

- Individual organization members can be made into an organization admin
- Organization admins have full admin access to all repositories in that organization
- Can add users to the organization
- Can create and configure teams in the organization

DEMO & EXERCISE: DTR TEAMS

Instructor will demonstrate creating organizations and teams in DTR and the access control associated with them via the 'Working with Organizations and Teams' exercise.

QUIZ

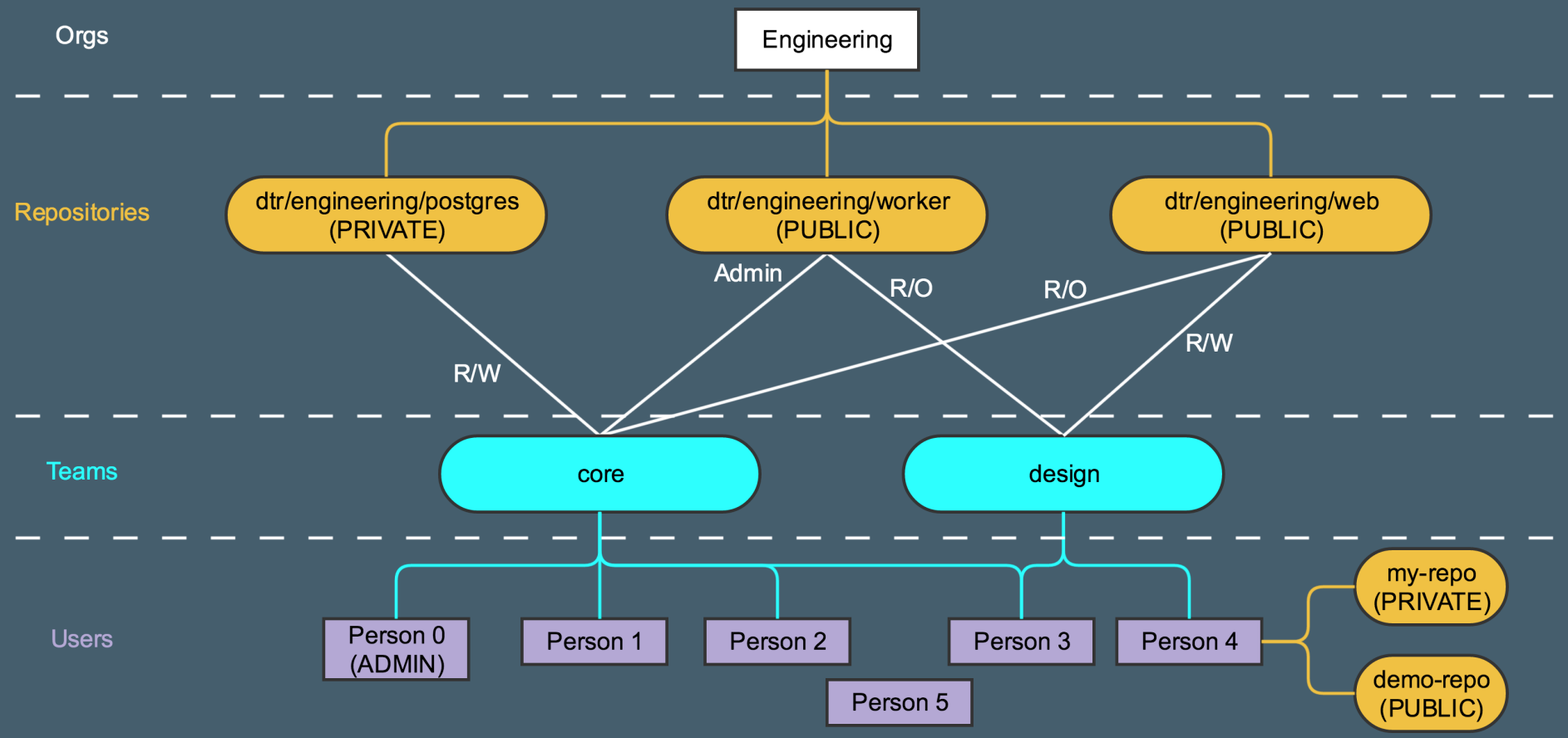
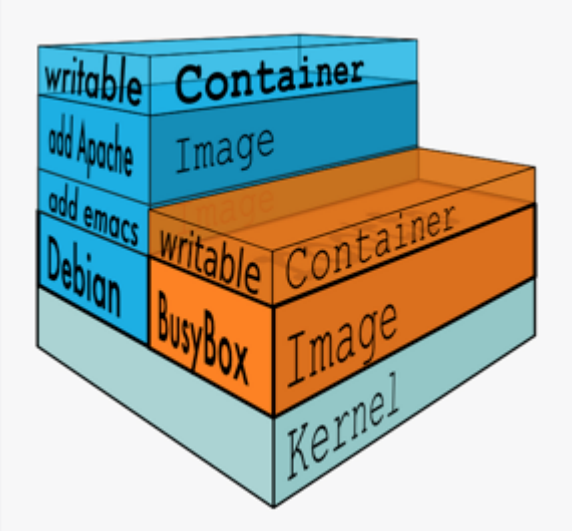

Orgs
Engineering
Repositories
dtr/engineering/postgres
(PRIVATE)
dtr/engineering/worker
(PUBLIC)
dtr/engineering/web
(PUBLIC)
Admin
R/O
R/O
R/W
R/W
Teams
core
design
Users
Person 0
(ADMIN)
Person 1
Person 2
Person 3
Person 4
Person 5
my-repo
(PRIVATE)
demo-repo
(PUBLIC)

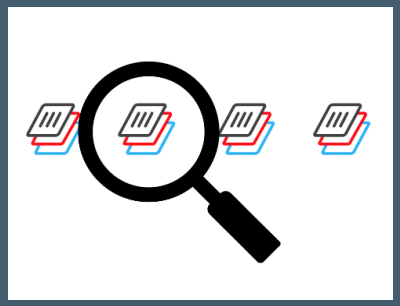
IMAGE SECURITY SCANNING

LAYERED FILESYSTEMS & PROACTIVE SECURITY



- Images made of read-only (shared) layers
- Possible to scan each layer and produce a manifest of what's inside
- Regularly check manifests against CVE databases
- Provide webhook triggers to offer security in CI/CD

DTR SECURITY SCAN FEATURES



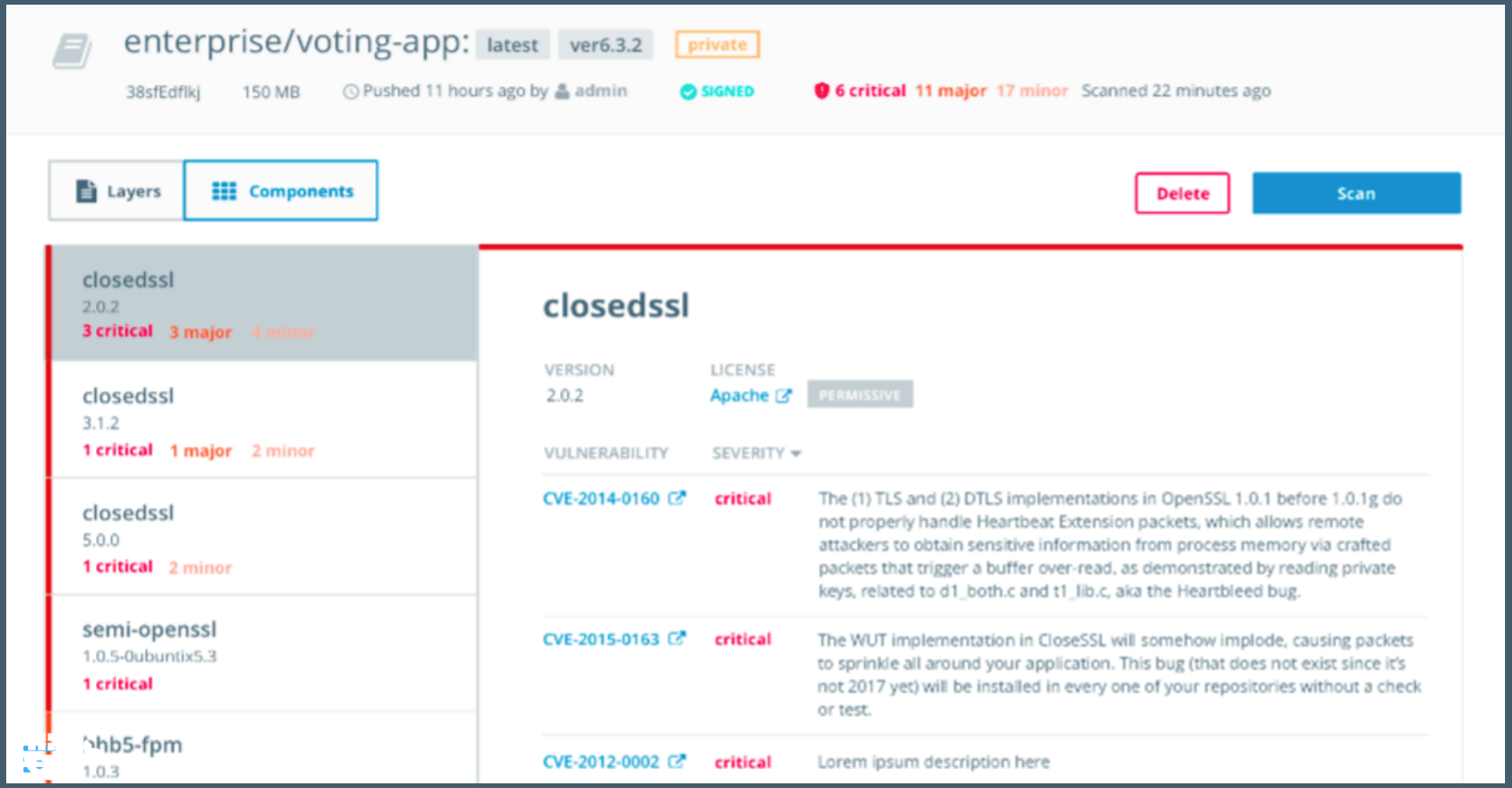
- Layers: Integrated security scanning and vulnerability monitoring
- Components: Binary level scanning provides deep visibility into components
- Continuous vulnerability monitoring with ability to customize alerts, policies and configurations.

LAYER VULNERABILITY MONITORING

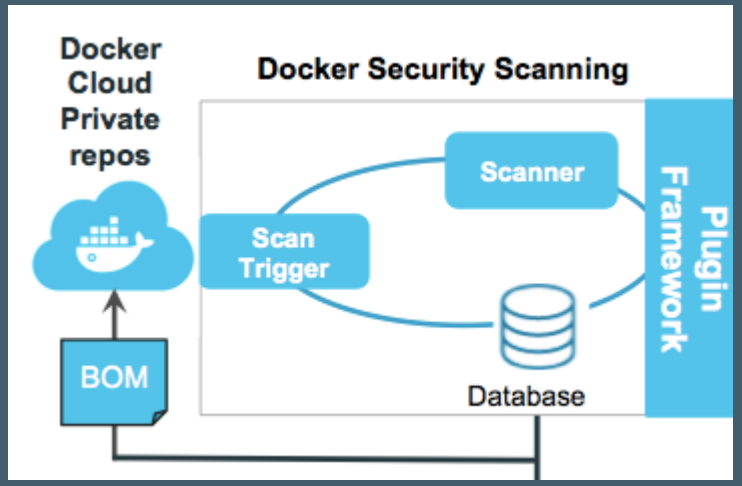
Full BOM for a Docker Image

COMPONENT VULNERABILITIES AND LICENSING

Binary level scanning provides deep visibility into components

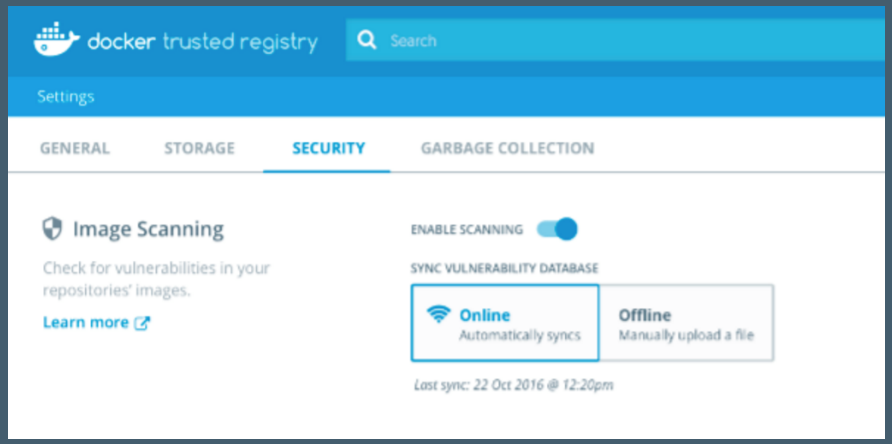


HOW DTR SECURITY SCANNING WORKS



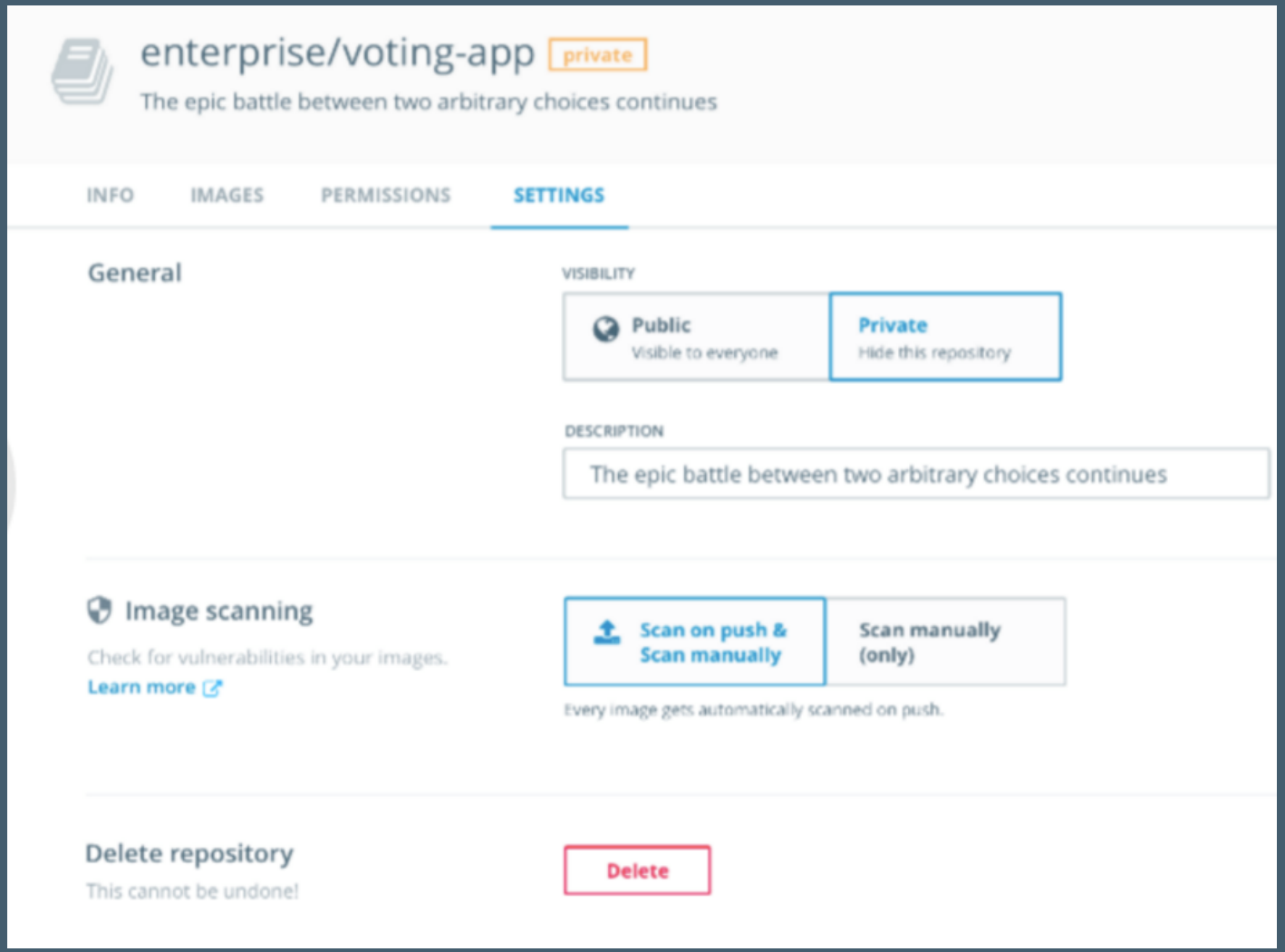
- Image layers get scanned individually
- Two phases:
  - scan (Bill-of-Materials generation)
  - check (checking the BoM against the vulnerability database)
- Store the BoM for each layer
- Longer to scan and shorter to check
- Regular database updates from https://dss-cve-updates.docker.com/

CVE DATABASE UPDATES



- DTR -> Settings -> Security
- Automatic updates: daily 3 AM UTC; must be able to reach https://dss-cve-updates.docker.com/ on port 443.
- Manual updates: uploaded through DTR, request from nautilus-feedback@docker.com

SCANNING AUTOMATION



EXERCISE: IMAGE SCANNING

Work through the 'Image Scanning in DTR' exercise in the Docker for Enterprise Operations Exercise book.

DTR WEBHOOKS

DTR WEBHOOKS

POST message with JSON payload, triggered on:

- Repo events: `TAG_PUSH`, `TAG_DELETE`, `MANIFEST_PUSH`, `MANIFEST_DELETE`, `SCAN_COMPLETED`, `SCAN_FAILED`
- Namesapce events (repo CRUD): `REPO_EVENT`
- Global events: `SCANNER_UPDATE_COMPLETED`

SETTING UP A WEBHOOK

1. Navigate to the API Docs through DTR's top-right user menu
2. Open the `POST api/v0/webhooks` form
3. Enter the webhook's metadata:

```
{
  "type": "TAG_PUSH",
  "key": "myorg/repo-of-interest",
  "endpoint": "http://url-to-post-hook-to:5000"
}
```

4. Click 'Try it Out!' to register webhook with DTR
5. Click 'Try it Out!' on the GET request for `api/v0/webhooks` to list all registered webhooks.

WEBHOOK PAYLOAD

- Webhook payloads always come in a wrapper:

```
{
  "type": "...",
  "createdAt": "2012-04-23T18:25:43.511Z",
  "contents": {...}
}
```

- The `contents` key depends on the event type; see http://dockr.ly/2knbu3J for the full spec.

EXERCISE: DTR WEBHOOKS

Work through the 'DTR Webhooks' exercise in the Docker for Enterprise Operations Exercise book.

INTRODUCTION TO ENTERPRISE DOCKER OPERATIONS TAKEAWAYS

In this workshop, we explored:

- Universal Control Plane
  - Application management
  - User management
  - Datacenter architecture
- Docker Trusted Registry
  - Image sharing
  - Image scanning
  - Webhooks
- ...plus other odds and ends.

All of which focus on provisioning an enterprise-grade **software supply chain**.


DOCKERCON 2017 - AUSTIN
Thanks for coming! Please take our feedback survey:
https://dockertraining.typeform.com/to/DeSKFW
Get in touch: training@docker.com
training.docker.com