TOPICS

- Key Software
TOPICS

- Key Software
- OpenShift Builds: Broad Strokes
TOPICS

- Key Software
- OpenShift Builds: Broad Strokes
- 3.11: A (Relatively) Simpler Time
TOPICS

- Key Software
- OpenShift Builds: Broad Strokes
- 3.11: A (Relatively) Simpler Time
- The Road to Using CRI
TOPICS

- Key Software
- OpenShift Builds: Broad Strokes
- 3.11: A (Relatively) Simpler Time
- The Road to Using CRI
- 4.0: First Pass
TOPICS

- Key Software
- OpenShift Builds: Broad Strokes
- 3.11: A (Relatively) Simpler Time
- The Road to Using CRI
- 4.0: First Pass
- Stuff We Broke: A Partial List
TOPICS

- Key Software
- OpenShift Builds: Broad Strokes
- 3.11: A (Relatively) Simpler Time
- The Road to Using CRI
- 4.0: First Pass
- Stuff We Broke: A Partial List
- 4.0: Fixes and Finish
KEY SOFTWARE
KUBERNETES
KUBERNETES

- An open-source system for automating deployment, scaling, and management of containerized applications
KUBERNETES

- An open-source system for automating deployment, scaling, and management of containerized applications
- Controlled over gRPC, for example by `kubectl`
KUBERNETES

- An open-source system for automating deployment, scaling, and management of containerized applications
- Controlled over gRPC, for example by kubectl
- Kind of a big deal
OPENSHIFT
OPENSSHIFT

- Based on Kubernetes
OPENSHIFT

- Based on Kubernetes
- Controlled over gRPC, for example by `kubectl` or `oc`
OPENSHIFT

- Based on Kubernetes
- Controlled over gRPC, for example by kubectl or oc
- Application Lifecycle Management
OPENSHIFT

- Based on Kubernetes
- Controlled over gRPC, for example by `kubectl` or `oc`
- Application Lifecycle Management
- DevOps Tooling
OPENSHIFT

- Based on Kubernetes
- Controlled over gRPC, for example by `kubectl` or `oc`
- Application Lifecycle Management
- DevOps Tooling
- Builds!
DOCKERD
DOCKERD

- Downloads images, runs containers
DOCKERD

- Downloads images, runs containers
- Controlled using its REST API
DOCKERD

- Downloads images, runs containers
- Controlled using its REST API
- REST API includes a Build endpoint
DOCKER BUILD
DOCKER BUILD

- Uses the dockerd REST API to transfer the build context as a tarball
DOCKER BUILD

- Uses the dockerd REST API to transfer the *build context* as a tarball
- Image is built completely "remotely" by the engine
DOCKER BUILD

- Uses the dockerd REST API to transfer the build context as a tarball
- Image is built completely "remotely" by the engine
- Engine parses a Dockerfile included in the build context to determine build steps
DOCKER BUILD

- Uses the dockerd REST API to transfer the *build context* as a tarball
- Image is built completely "remotely" by the engine
- Engine parses a Dockerfile included in the *build context* to determine build steps
- Build process is directed in-engine
DOCKER BUILD

- Uses the dockerd REST API to transfer the build context as a tarball
- Image is built completely "remotely" by the engine
- Engine parses a Dockerfile included in the build context to determine build steps
- Build process is directed in-engine
- Base image is pulled into engine's storage area
DOCKER BUILD

- Uses the dockerd REST API to transfer the build context as a tarball
- Image is built completely "remotely" by the engine
- Engine parses a Dockerfile included in the build context to determine build steps
- Build process is directed in-engine
- Base image is pulled into engine's storage area
- New image is saved alongside base image in engine's storage area
IMAGEBUILDER

- Parses Dockerfile to determine build steps
IMAGEBUILDER

- Parses Dockerfile to determine build steps
- Executes build steps using callbacks
- Provides callbacks that use go-dockerclient to talk to a remote engine
IMAGEBUILDER

- Parses Dockerfile to determine build steps
- Executes build steps using callbacks
- Provides callbacks that use go-dockerclient to talk to a remote engine
- A library and a CLI wrapper
SOURCE-TO-IMAGE

- Uses one of many suitable builder images
SOURCE-TO-IMAGE

- Uses one of many suitable builder images
- Builder images use information in your source tree to guide build steps
SOURCE-TO-IMAGE

- Uses one of many suitable builder images
- Builder images use information in your source tree to guide build steps
- Uses docker client library to start a builder image using a remote engine
SOURCE-TO-IMAGE

- Uses one of many suitable builder images
- Builder images use information in your source tree to guide build steps
- Uses docker client library to start a builder image using a remote engine
- Uses docker client library to commit the builder container using the remote engine
SOURCE-TO-IMAGE

- Uses one of many suitable builder images
- Builder images use information in your source tree to guide build steps
- Uses docker client library to start a builder image using a remote engine
- Uses docker client library to commit the builder container using the remote engine
- As of 1.1.11, can be told to produce a Dockerfile instead of using a remote engine
**SOURCE-TO-IMAGE**

- Uses one of many suitable builder images
- Builder images use information in your source tree to guide build steps
- Uses docker client library to start a builder image using a remote engine
- Uses docker client library to commit the builder container using the remote engine
- As of 1.1.11, can be told to produce a Dockerfile instead of using a remote engine
- A library and a CLI wrapper
OPENSHIFT-**-BUILDER

- Traditionally, the oc binary invoked under a different name
Traditionally, the `oc` binary invoked under a different name
- `openshift-docker-builder`
  - multi-layer builds
    - runs Dockerfile builds in `dockerd`
    - using `go-dockerclient`
    - to call `dockerd`'s build API
  - single-layer builds
    - runs Dockerfile builds in `dockerd`
    - using `imagebuilder` as a library
    - using callbacks that ultimately call `go-dockerclient`
    - to call `dockerd`'s `run/add/copy/commit` APIs
OPENSHIFT-**-BUILDER

- Traditionally, the `oc` binary invoked under a different name
- openshift-docker-builder
  - multi-layer builds
    - runs Dockerfile builds in `dockerd`
    - using `go-dockerclient`
    - to call `dockerd`'s build API
  - single-layer builds
    - runs Dockerfile builds in `dockerd`
    - using `imagebuilder` as a library
    - using callbacks that ultimately call `go-dockerclient`
    - to call `dockerd`'s run/add/copy/commit APIs
- openshift-sti-builder
  - runs source-to-image builds using `dockerd` using the `docker` client package
OPENSHEET BUILDS

The Broad Strokes
BUILD TRIGGERS

- Webhooks
  - bitbucket
  - generic
  - github
  - gitlab
BUILD TRIGGERS

- Webhooks
  - bitbucket
  - generic
  - github
  - gitlab
- API calls (`oc new-build` or editing the BuildConfig)
BUILD TRIGGERS

- Webhooks
  - bitbucket
  - generic
  - github
  - gitlab
- API calls (oc new-build or editing the BuildConfig)
- Other images being updated
THE BUILD POD

- Runs the main builder image: openshift-docker-build or openshift-sti-build
THE BUILD POD

- Optionally downloads sources to a volume directory
- Runs the main builder image: `openshift-docker-build` or `openshift-sti-build`
THE BUILD POD

- Optionally downloads sources to a volume directory
- Runs the main builder image: `openshift-docker-build` or `openshift-sti-build`
  - If sources have been downloaded, they're in a mounted volume
THE BUILD POD

- Optionally downloads sources to a volume directory
- Runs the main builder image: `openshift-docker-build` or `openshift-sti-build`
  - If sources have been downloaded, they're in a mounted volume
  - Has to build an image, somehow
    - Usually that means dockerd's build or run/commit APIs
THE BUILD POD

- Optionally downloads sources to a volume directory
- Runs the main builder image: `openshift-docker-build` or `openshift-sti-build`
  - If sources have been downloaded, they're in a mounted volume
  - Has to build an image, somehow
    - Usually that means `dockerd`'s build or run/commit APIs
- Has to push the image to a registry, somehow
  - Usually that means `dockerd`'s push API
THE BUILD POD

- Optionally downloads sources to a volume directory
- Runs the main builder image: `openshift-docker-build` or `openshift-sti-build`
  - If sources have been downloaded, they're in a mounted volume
  - Has to build an image, somehow
    - Usually that means `dockerd`'s build or run/commit APIs
- Has to push the image to a registry, somehow
  - Usually that means `dockerd`'s push API
- Indicates success or failure
REDEPLOYMENT
REDPLOYMENT

- New images get noticed
  - A tag is changed to point to a different version
  - The DeploymentConfig is itself modified
REDEPLOYMENT

- New images get noticed
  - A tag is changed to point to a different version
  - The DeploymentConfig is itself modified
- Pods that used old images are restarted using new images
  - Recreate all affected pods
  - Rolling updates
3.11: A (RELATIVELY) SIMPLER TIME
THE BUILD CONTROLLER
THE BUILD CONTROLLER

- Reads the build config to determine
  - Whether it’s a Dockerfile or source-to-image build (the *strategy*)
  - Where the source code lives
    - a git repository location
    - the contents of a Dockerfile
    - a binary file
    - in one or more other images
  - What image name to push to
THE BUILD CONTROLLER

- Reads the build config to determine
  - Whether it's a Dockerfile or source-to-image build (the strategy)
  - Where the source code lives
    - a git repository location
    - the contents of a Dockerfile
    - a binary file
    - in one or more other images
  - What image name to push to
- Specifies the build pod to run
  - Optionally download the sources into a volume
    - an "init" container
  - Dockerfile strategy: pod runs openshift-docker-builder
  - Source-to-image strategy: pod runs openshift-sti-builder
  - Custom strategy: pod runs a container based on... an image
AN EXAMPLE BUILDCONFIG

apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
  annotations:
    openshift.io/generated-by: OpenShiftNewApp
  creationTimestamp: 2019-01-26T14:38:35Z
labels:
  app: ruby-ex
name: ruby-ex
namespace: example
resourceVersion: "442235"
selfLink: /apis/build.openshift.io/v1/namespaces/example/buildconfigs/ruby-ex
uid: 2fe2bf6e-20ce-11e9-a8b5-0a580a80001a
apiVersion: v1
kind: Pod
metadata:
  annotations:
    openshift.io/build.name: ruby-ex-1
    openshift.io/scc: privileged
creationTimestamp: 2019-01-26T16:38:35Z
labels:
  openshift.io/build.name: ruby-ex-1
name: ruby-ex-1-build
namespace: example
ownerReferences:
- apiVersion: build.openshift.io/v1
THE ROAD TO USING CRI
LET ME SUM UP
LET ME SUM UP

- A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
LET ME SUM UP

- A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
- A desire to be able to inspect images in registries leads to skopeo becoming a thing.
LET ME SUM UP

- A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
- A desire to be able to inspect images in registries leads to skopeo becoming a thing.
- Skopeo gets refactored into `containers/image` and a CLI wrapper.
LET ME SUM UP

- A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
- A desire to be able to inspect images in registries leads to skopeo becoming a thing.
- Skopeo gets refactored into *containers/image* and a CLI wrapper.
- *runc*, a CLI tool for spawning and running containers according to the OCI specification, becomes a thing.
LET ME SUM UP

- A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
- A desire to be able to inspect images in registries leads to skopeo becoming a thing.
- Skopeo gets refactored into containers/image and a CLI wrapper.
- runc, a CLI tool for spawning and running containers according to the OCI specification, becomes a thing.
- CRI-O, using containers/image to pull down images and runc to run containers, becomes a thing.
LET ME SUM UP

- A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
- A desire to be able to inspect images in registries leads to skopeo becoming a thing.
- Skopeo gets refactored into containers/image and a CLI wrapper.
- runc, a CLI tool for spawning and running containers according to the OCI specification, becomes a thing.
- CRI-O, using containers/image to pull down images and runc to run containers, becomes a thing.
- containers/storage becomes a thing, and then becomes a thing that CRI-O uses.
A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
A desire to be able to inspect images in registries leads to skopeo becoming a thing.
Skopeo gets refactored into containers/image and a CLI wrapper.
runc, a CLI tool for spawning and running containers according to the OCI specification, becomes a thing.
CRI-O, using containers/image to pull down images and runc to run containers, becomes a thing.
containers/storage becomes a thing, and then becomes a thing that CRI-O uses.
OpenShift gets interested in running on top of CRI-O.
A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
A desire to be able to inspect images in registries leads to skopeo becoming a thing.
Skopeo gets refactored into containers/image and a CLI wrapper.
runc, a CLI tool for spawning and running containers according to the OCI specification, becomes a thing.
CRI-O, using containers/image to pull down images and runc to run containers, becomes a thing.
containers/storage becomes a thing, and then becomes a thing that CRI-O uses.
OpenShift gets interested in running on top of CRI-O.
OpenShift needs to be able to build, but CRI doesn't provide methods for creating new images.
  - No build for building images outright.
  - No commit for driving a build from a client.
A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
A desire to be able to inspect images in registries leads to skopeo becoming a thing.
Skopeo gets refactored into containers/image and a CLI wrapper.
runc, a CLI tool for spawning and running containers according to the OCI specification, becomes a thing.
CRI-O, using containers/image to pull down images and runc to run containers, becomes a thing.
containers/storage becomes a thing, and then becomes a thing that CRI-O uses.
OpenShift gets interested in running on top of CRI-O.
OpenShift needs to be able to build, but CRI doesn't provide methods for creating new images.
  - No build for building images outright.
  - No commit for driving a build from a client.
buildah becomes a thing, gets refactored into a library and a CLI wrapper.
A desire to support multiple runtime engines leads to Kubernetes adopting CRI.
A desire to be able to inspect images in registries leads to skopeo becoming a thing.
Skopeo gets refactored into containers/image and a CLI wrapper.
runc, a CLI tool for spawning and running containers according to the OCI specification, becomes a thing.
CRI-O, using containers/image to pull down images and runc to run containers, becomes a thing.
containers/storage becomes a thing, and then becomes a thing that CRI-O uses.
OpenShift gets interested in running on top of CRI-O.
OpenShift needs to be able to build, but CRI doesn't provide methods for creating new images.
- No build for building images outright.
- No commit for driving a build from a client.
buidlah becomes a thing, gets refactored into a library and a CLI wrapper.
We decide to look at replacing remote calls in the builder images with calls into buildah.
4.0: FIRST PASS
BUILDER IMAGE UPDATES

- openshift-docker-build
  - switches from using client calls to wrapping calls to buildah as a library
BUILDER IMAGE UPDATES

- openshift-docker-build
  - switches from using client calls to wrapping calls to buildah as a library
- openshift-sti-build
  - switches to producing Dockerfiles
  - feeds the Dockerfiles to buildah as a library
BUilder image updates

- openshift-docker-build
  - switches from using client calls to wrapping calls to buildah as a library
- openshift-sti-build
  - switches to producing Dockerfiles
  - feeds the Dockerfiles to buildah as a library
- Image builds now happen inside the build container. Builds are no longer remote
**BUILDER IMAGE UPDATES**

- openshift-docker-build
  - switches from using client calls to wrapping calls to buildah as a library
- openshift-sti-build
  - switches to producing Dockerfiles
  - feeds the Dockerfiles to buildah as a library
- Image builds now happen inside the build container. **Builds are no longer remote**
- The build container pushes image content to registries from inside itself
STUFF WE BROKE

A Partial List
NEW DEPENDENCIES
NEW DEPENDENCIES

- Like, a lot of dependencies
NEW DEPENDENCIES

- Like, a lot of dependencies
- Many of them Linux-specific
NEW DEPENDENCIES

- Like, a lot of dependencies
- Many of them Linux-specific
- `oc` wants to be small and needs to run on many platforms
- `openshift-*-builder` runs in a container, so it only needs to run on Linux
NEW DEPENDENCIES

- Like, a lot of dependencies
- Many of them Linux-specific
- `oc` wants to be small and needs to run on many platforms
- `openshift-*-builder` runs in a container, so it only needs to run on Linux
- And so `openshift/builder` became a separate repository
BUILD IN A CONTAINER
BUILD IN A CONTAINER

- Cached layers aren't shared between builds
BUILD IN A CONTAINER

- Cached layers aren't shared between builds
- Cached layers aren't available when a failed build is retried
  - A retry still has to run all of the steps that succeeded during previous attempts
BUILD IN A CONTAINER

- Cached layers aren't shared between builds
- Cached layers aren't available when a failed build is retried
  - A retry still has to run all of the steps that succeeded during previous attempts
- Images pushed from a container to a registry aren't instantly available on the host
  - Not sharing storage means the host still needs to pull the new image to run it
BUILD IN A CONTAINER
BUILD IN A CONTAINER

- A container that's pushing an image doesn't know which CAs the host trusts
  - And the CAs the host trusts might not be the right set, either
BUILD IN A CONTAINER

- A container that's pushing an image doesn't know which CAs the host trusts
  - And the CAs the host trusts might not be the right set, either
- Recompressing base image / builder image layers can waste time and disk space
BUILD IN A CONTAINER

- A container that’s pushing an image doesn't know which CAs the host trusts
  - And the CAs the host trusts might not be the right set, either
- Recompressing base image / builder image layers can waste time and disk space
- A container that’s pushing an image doesn't know which cross-repository mounts to try to use
BUILD IN A CONTAINER

- A container that's pushing an image doesn't know which CAs the host trusts
  - And the CAs the host trusts might not be the right set, either
- Recompressing base image / builder image layers can waste time and disk space
- A container that's pushing an image doesn't know which cross-repository mounts to try to use
- Builds are still run privileged
4.0: FIXES AND FINISH
NO NODE CONFIG
NO NODE CONFIG

- OpenShift 4 is opinionated. It's got takes
NO NODE CONFIG

- OpenShift 4 is opinionated. It's got takes
- Can't depend on the node's configuration
NO NODE CONFIG

- OpenShift 4 is opinionated. It’s got takes
- Can’t depend on the node’s configuration
- Can’t foist configuration problems onto the node and its administrator
NO NODE CONFIG

- OpenShift 4 is opinionated. It's got takes
- Can't depend on the node's configuration
- Can't foist configuration problems onto the node and its administrator
- It's all in configmaps with reasonable defaults
WHAT'S NEXT
WHAT'S NEXT

- Bug fixes
  - Multilayer builds
  - Search registries for completing base image names that don't include registry parts
WHAT'S NEXT

- Bug fixes
  - Multilayer builds
  - Search registries for completing base image names that don't include registry parts
- Unprivileged builds
WHAT'S NEXT

- Bug fixes
  - Multilayer builds
  - Search registries for completing base image names that don't include registry parts
- Unprivileged builds
- Experimental things
WHAT'S NEXT

- Bug fixes
  - Multilayer builds
  - Search registries for completing base image names that don't include registry parts
- Unprivileged builds
- Experimental things
- Pretty sure you have some ideas
POINTERS TO PACKAGES

- Kubernetes: http://kubernetes.io
- OpenShift: http://openshift.org
- Docker: https://www.docker.com/products/docker-engine
- go-dockerclient: https://github.com/fsouza/go-dockerclient
- imagebuilder: https://github.com/openshift/imagebuilder
- source-to-image: https://github.com/openshift/source-to-image
- skopeo: https://github.com/containers/skopeo
- containers/image: https://github.com/containers/image
- runc: https://github.com/opencontainers/runc
- CRI-O: https://github.com/kubernetes-sigs/cri-o
- containers/storage: https://github.com/containers/storage
- buildah: https://github.com/containers/buildah
- openshift/builder: https://github.com/openshift/builder/
QUESTIONS
THANK YOU