Agenda

Image Layout: Stop Putting Everything in Registries

Brandon Mitchell
Twitter: @sudo_bmitch
GitHub: sudo-bmitch

```
$ whoami
- Brandon Mitchell
- Solutions Architect @ BoxBoat an IBM Company
- Docker Captain
- StackOverflow, OCI, CNCF
- Software Supply Chain Security
```

@sudo_bmitch

Container Images

Image Config

```
$ regctl blob get alpine sha256:e66264b987... | jq .
{
  "architecture": "amd64",
  "config": {
    "User": "",
    "Env": [
      "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
    ],
    "Cmd": [
      "/bin/sh"
    ],
    "Volumes": null,
    "WorkingDir": "",
    "Entrypoint": null,
    "OnBuild": null,
    "Labels": null
...
```

Content Addressable

```
$ regctl blob get alpine sha256:e66264b9877... | sha256sum
e66264b9877...  -
```

Image Layers

```
$ regctl blob get alpine sha256:2408cc74d1... | tar -tvzf -
drwxr-xr-x 0/0               0 2022-05-23 12:51 bin/
lrwxrwxrwx 0/0               0 2022-05-23 12:51 bin/arch -> /bin/busybox
lrwxrwxrwx 0/0               0 2022-05-23 12:51 bin/ash -> /bin/busybox
lrwxrwxrwx 0/0               0 2022-05-23 12:51 bin/base64 -> /bin/busybox
lrwxrwxrwx 0/0               0 2022-05-23 12:51 bin/bbconfig -> /bin/busybox
-rwxr-xr-x 0/0          837272 2022-05-09 13:27 bin/busybox
lrwxrwxrwx 0/0               0 2022-05-23 12:51 bin/cat -> /bin/busybox
lrwxrwxrwx 0/0               0 2022-05-23 12:51 bin/chattr -> /bin/busybox
lrwxrwxrwx 0/0               0 2022-05-23 12:51 bin/chgrp -> /bin/busybox
lrwxrwxrwx 0/0               0 2022-05-23 12:51 bin/chmod -> /bin/busybox
...
```

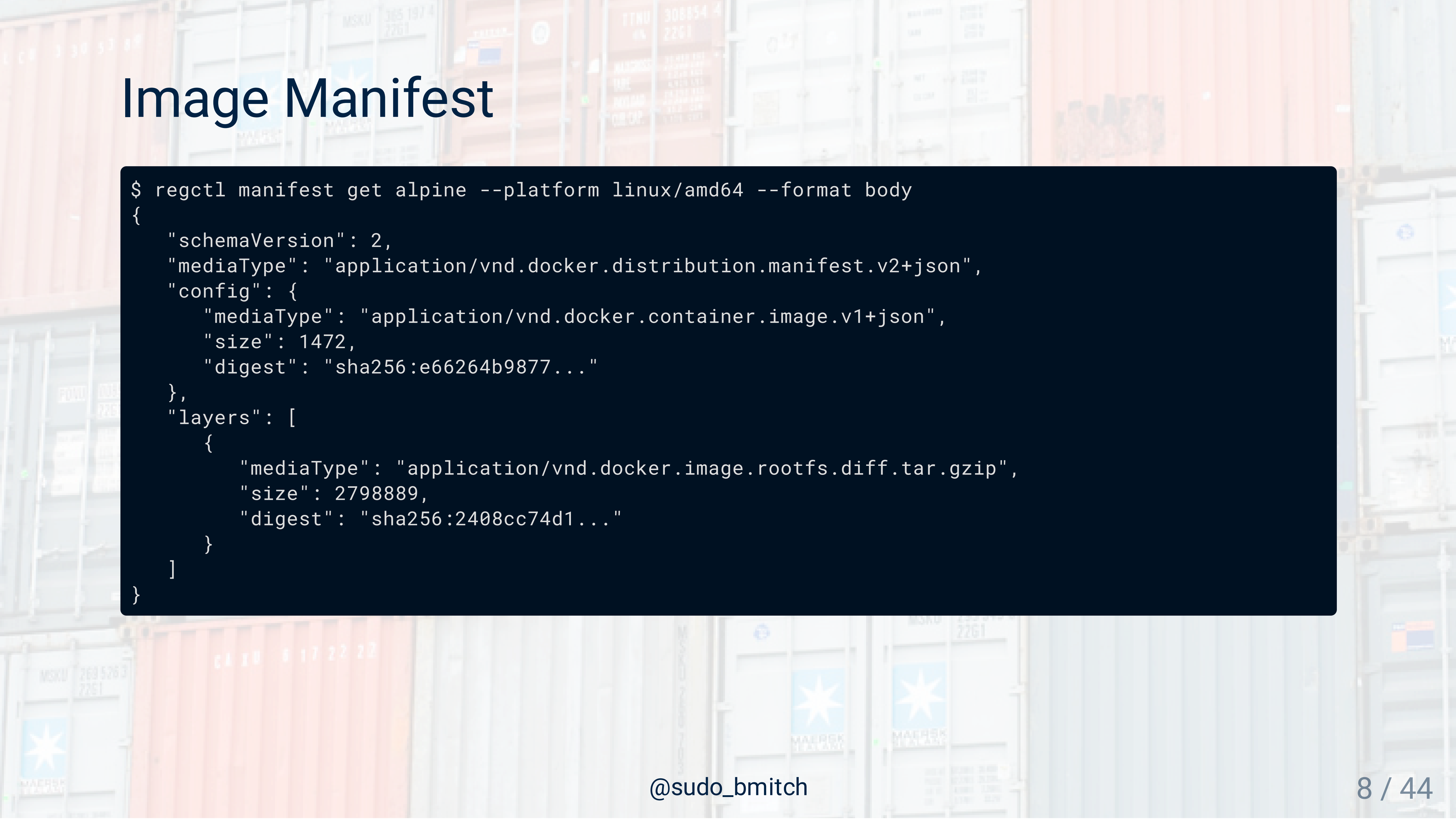
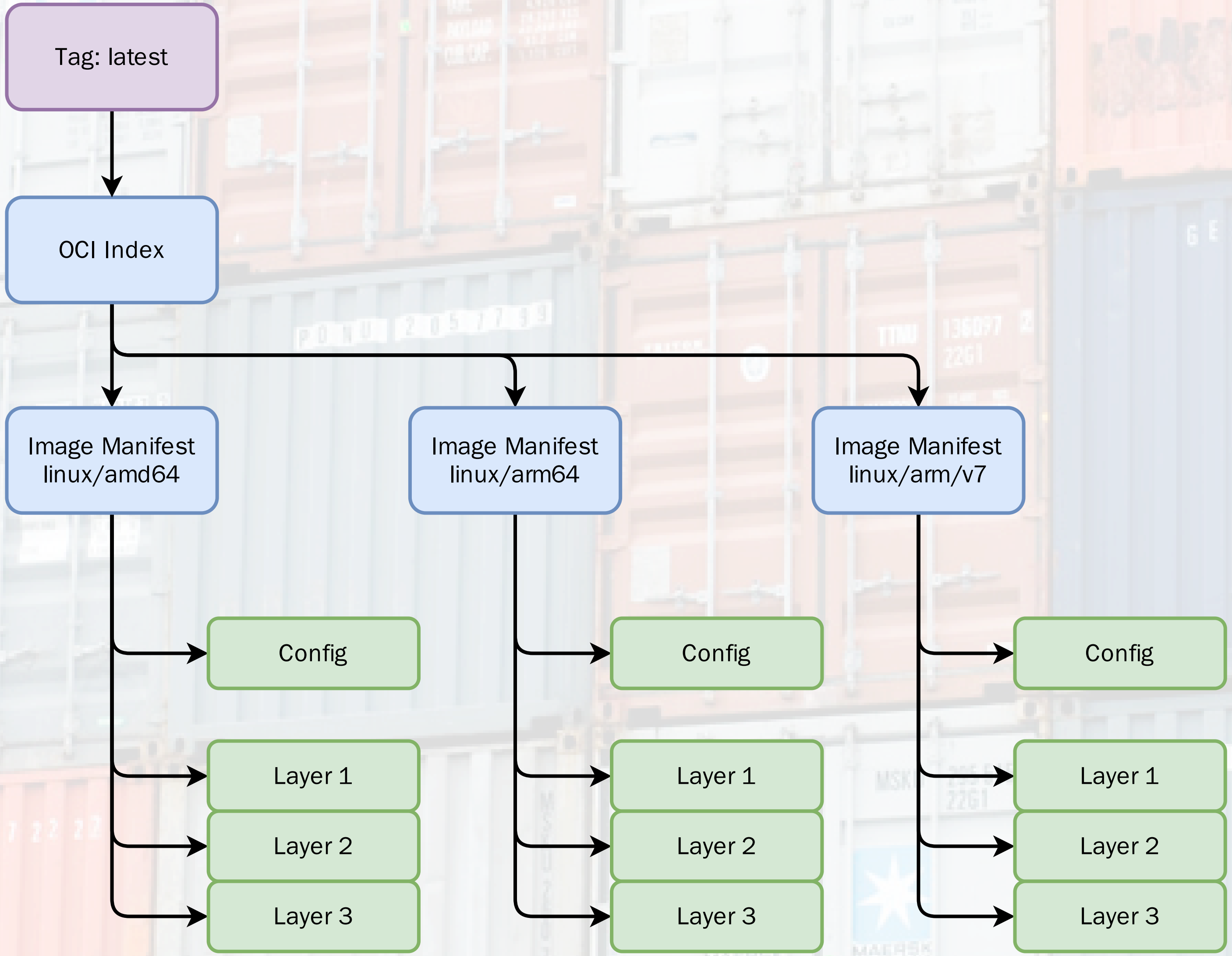


Image Manifest

```
$ regctl manifest get alpine --platform linux/amd64 --format body
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
   "config": {
      "mediaType": "application/vnd.docker.container.image.v1+json",
      "size": 1472,
      "digest": "sha256:e66264b9877..."
   },
   "layers": [
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 2798889,
         "digest": "sha256:2408cc74d1..."
      }
   ]
}
```

Multi-Platform Manifest

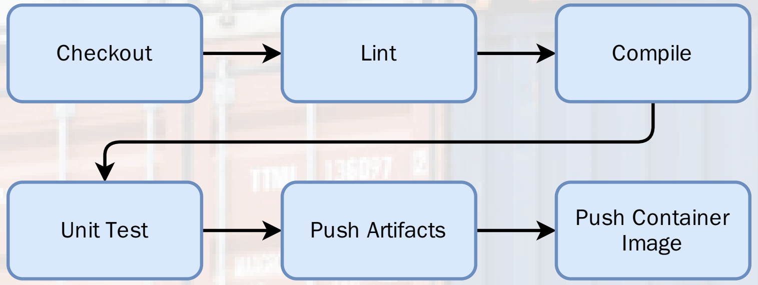
```
$ regctl manifest get alpine --format body | jq .
{
  "manifests": [
    {
      "digest": "sha256:4ff3ca9127...",
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      },
      "size": 528
    },
    {
      "digest": "sha256:3c66139adb...",
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "platform": {
        "architecture": "arm",
        "os": "linux",
        "variant": "v6"
      },
      "size": 528
...
```


Tag: latest
OCI Index
Image Manifest linux/amd64
Image Manifest linux/arm64
Image Manifest linux/arm/v7
Config
Layer 1
Layer 2
Layer 3
Config
Layer 1
Layer 2
Layer 3
Config
Layer 1
Layer 2
Layer 3

CI/CD Pipelines

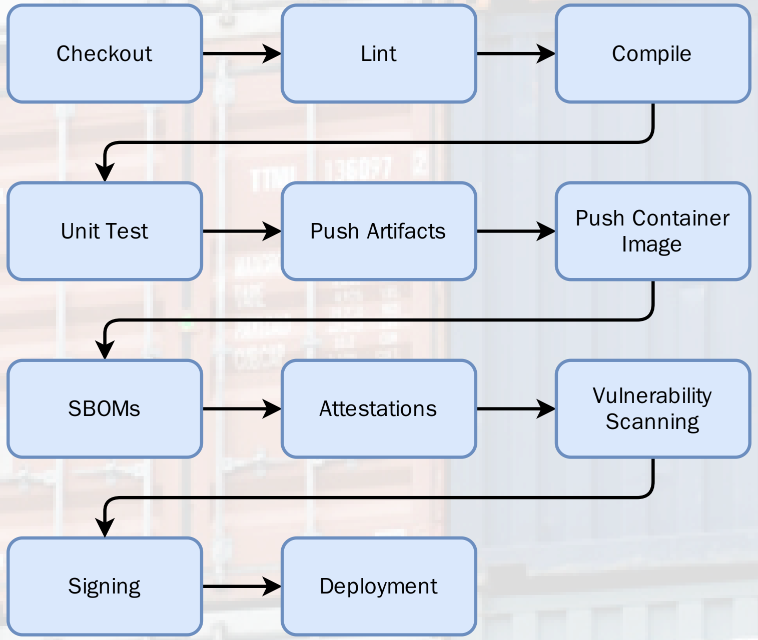
CI Pipelines

- More than Build, Ship, Run
- Linting
- Compile
- Unit tests
- Push artifacts
- Push container images
- Integration tests



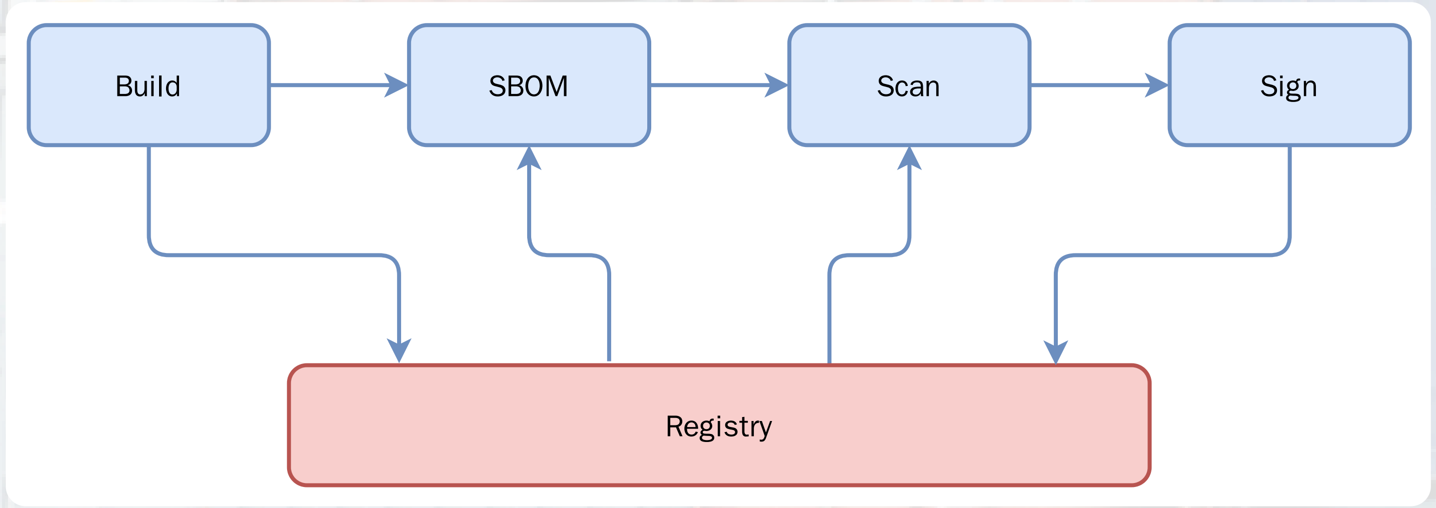
But Wait, There's More

- SBOMs
- Signing
- Attestations
- Vulnerability scanning
- Cat pics



Should We Use a Registry?

- CI pipelines are designed to work with filesystems
- git clone -> lint -> compile -> unit test ...
- SBOM and vulnerability scanners pull the image layers from the registry


Build
SBOM
Scan
Sign
Registry

Storing Images On The Filesystem?

OCI Image Layout

```
$ tree .
├── oci-layout
├── index.json
└── blobs
    └── sha256
        ├── 1f9696fa2a7e90f098d7f5da0615b853989652ab5c597d37fb84d8038fff3acf
        ├── 2408cc74d12b6cd092bb8b516ba7d5e290f485d3eb9672efc00f0583730179e8
        ├── 29779da064c4b1c0d5d77ed112b35eed8fce7e2f35d2efea6ef68cb0d805d134
        ├── 44136fa355b3678a1146ad16f7e8649e94fb4fc21fe77e8310c060f61caaff8a
        ├── 63e1ceb3d310f9e355494e88bca9a522259a0de034f6dbd7b456994f4fb00265
        ├── 93d426e205de6adaec185e0c5bfe2435dc84aaf2b4a7d5bfeb33427a873516a2
        ├── 9783610725dd9ac5eace591938aa11ce36e3520581fb804dd874b58684297e8e
        ├── c223d141ab18a7a9ce6c8f43c1963aa29e9cbca57cefb5a96370929f6596ed2d
        ├── e3ab7e61a5ef77751e4ca958bc64210e11bf454bba6d757ce6b974a4677cc4ca
        ├── f1d9bcbd77f101ed0e5956bb84e68ddb03e39540de33e9d2e475c52ff851018c
        ├── f7af0f000e0feb1f2e439e7a1cff48d5bbe329abdcdd65bc07382236b38b98c9
        └── fe7f73a32c6ec78027e2e231f5c3063dc7a8ff5f2c67edb06e5626b996cfcaaa
```

OCI Layout

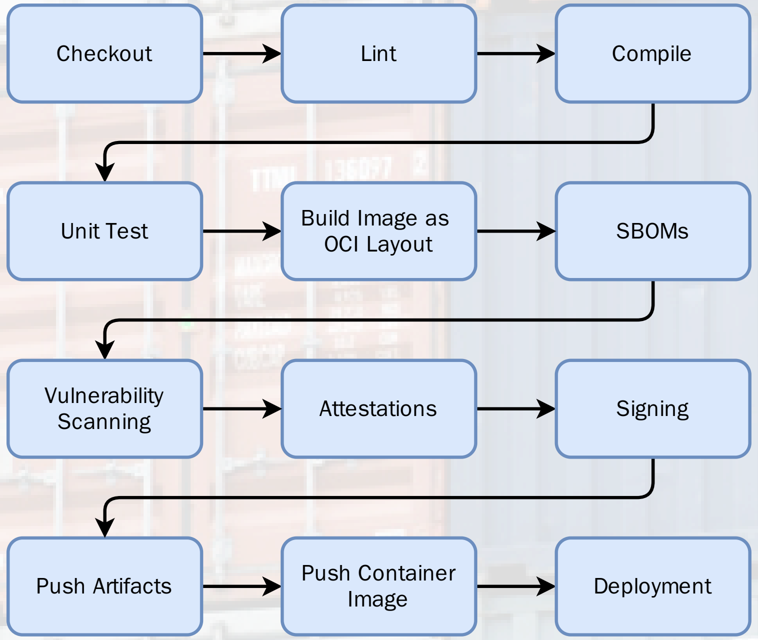
```
$ cat oci-layout
{"imageLayoutVersion":"1.0.0"}

$ cat index.json  | jq .
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.oci.image.index.v1+json",
  "manifests": [
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:f1d9bcbd77...",
      "size": 710,
      "annotations": {
        "org.opencontainers.image.ref.name": "latest"
      }
    }
  ]
}
```

Features

- Frequently used for air-gap environments
- Can be packaged as a tar or tgz for shipping
- Layout can coexist with `docker load` metadata

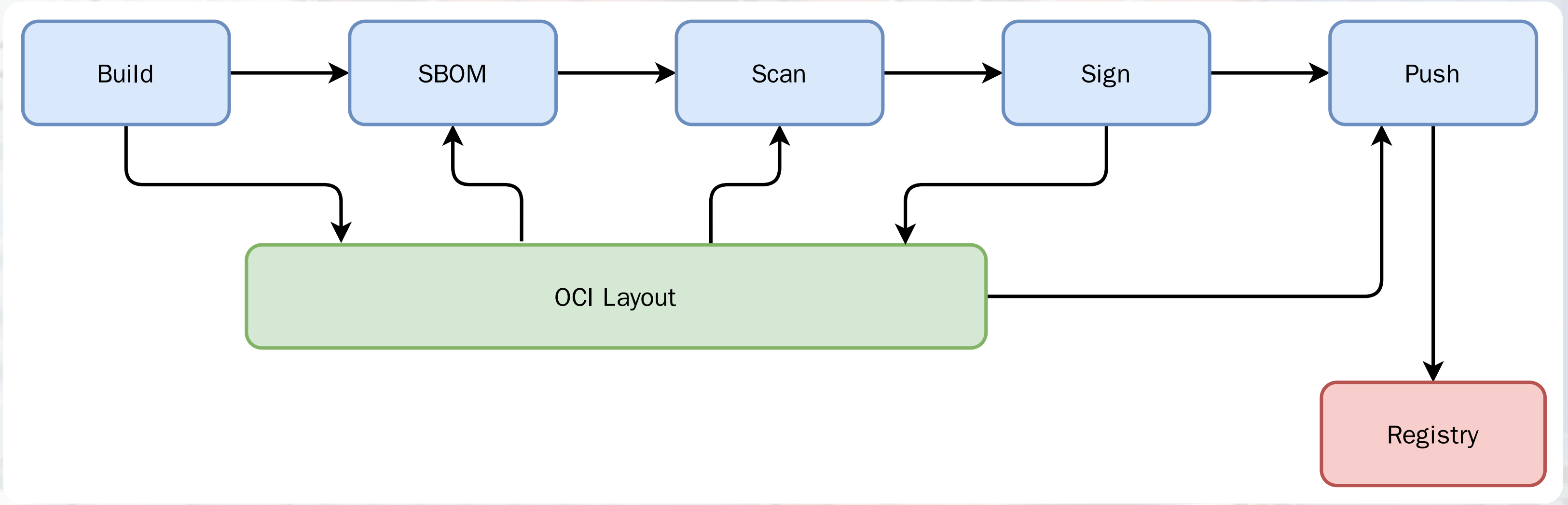
Pipeline of Tomorrow

- git clone
- Lint
- Image build -> OCI Layout
- Add Attestations
- Add SBOM
- Vulnerability scan
- Add signatures
- Add cat pictures
- Push to registry
- Deploy



Advantages

- Unscanned blobs are never pushed
- Unsigned images are never available to pull
- CI uses filesystem instead of network between steps


Build
SBOM
Scan
Sign
Push
OCI Layout
Registry

Sign Me Up

Some Tooling Supports This Today

- Runtimes: containerd, podman
- Build: docker buildx, buildah, buildpack, kaniko, ko
- Registries: zot
- Registry clients: crane, regclient, skopeo
- Vulnerability scanners: grype, snyk, trivy
- SBOM: syft, trivy
- Signing: ...

There Are Challenges

- References are not standardized
  - `alpine:latest` (becomes `docker.io/library/alpine:latest`)
  - `oci:dirname` (confused with `docker.io/library/oci:dirname`)
  - `oci-dir:dirname`
  - `ocidir://dirname:latest`
  - `oci-layout://dirname`
- OCI Working Group for Reference Types
- Garbage Collection

Lots Of Tools Do Not Support This Yet

- Some tools only work with registries
- Some tools only support tar/tgz packaged layouts
- Almost none support a layout with:
  - A specific tag
  - A specific digest
  - Multi-platform images

Zot Is A Workaround

```
$ regctl tag ls ocidir://oci-layout
latest
sha256-f1d9bcbd77....9783610725dd9ac5.sbom
sha256-f1d9bcbd77....c223d141ab18a7a9.scan

$ docker run -d -p 127.0.0.1:5207:5000 \
  -v "$(pwd)/oci-layout:/var/lib/registry/demo" \
  ghcr.io/project-zot/zot-linux-amd64:v1.4.0

$ cosign sign localhost:5207/demo:latest
...

$ regctl tag ls ocidir://oci-layout
latest
sha256-f1d9bcbd77....9783610725dd9ac5.sbom
sha256-f1d9bcbd77....c223d141ab18a7a9.scan
sha256-f1d9bcbd77....sig
```

Example

Setup Build Environment

```yaml
- name: Check out code
  uses: actions/checkout@v3

- name: Set up Go 1.17
  uses: actions/setup-go@v3
  with:
    go-version: 1.17.x
  id: go

- name: Install regctl
  uses: regclient/actions/regctl-installer@main

- name: Set up Docker Buildx
  uses: docker/setup-buildx-action@v1

- name: Install cosign
  uses: sigstore/cosign-installer@main
```

Test

```
- name: Verify go fmt
  run: test -z "$(go fmt ./...)"

- name: Verify go vet
  run: test -z "$(go vet ./...)"

- name: Test
  run: make test

- name: Linting
  run: make lint
```

Build to OCI Layout

```yaml
- name: Build
  uses: docker/build-push-action@v2
  id: build
  with:
    context: .
    # platforms: linux/amd64,linux/arm/v7,linux/arm64
    platforms: linux/amd64
    push: false
    labels: |
      org.opencontainers.image.source=${{ github.repositoryUrl }}
      org.opencontainers.image.revision=${{ github.sha }}
    outputs: |
      type=oci,dest=oci-layout.tar

- name: Convert to OCIDir
  run: regctl image import ocidir://oci-layout oci-layout.tar
```

Post-Build Tweaks

```
- name: Mutate
  run: |
    regctl image mod ocidir://oci-layout:latest --replace \
      --time-max "${{ steps.prep.outputs.date_commit }}" \
      --annotation "org.opencontainers.image.created=${{ steps.prep.outputs.date_commit }}" \
      --annotation "org.opencontainers.image.base.name=${{ steps.prep.outputs.base_name }}" \
      --annotation "org.opencontainers.image.base.digest=${{ steps.prep.outputs.base_digest }}" \
      --annotation "org.opencontainers.image.source=${{ github.repositoryUrl }}" \
      --annotation "org.opencontainers.image.revision=${{ github.sha }}"
```

Generate SBOM and Scan

```yaml
- name: SBOM
  uses: anchore/sbom-action@v0
  with:
    output-file: "sbom.json"
    image: "oci-dir:oci-layout"
    format: "cyclonedx-json"
- name: Scan
  uses: anchore/scan-action@v3
  with:
    image: "oci-dir:oci-layout"

- name: Attach artifacts to image
  id: artifacts
  run: |
    sbom_digest=$(regctl artifact put \
      --config-media-type application/vnd.oci.image.config.v1+json \
      -f sbom.json --media-type "application/vnd.cyclonedx+json" \
      --annotation org.opencontainers.artifact.type=sbom \
      --annotation org.example.sbom.type=cyclonedx-json \
      --format '{{ .Manifest.GetDescriptor.Digest }}' \
      --refers ocidir://oci-layout:latest)
    echo "::set-output name=sbom_digest::${sbom_digest}"
```

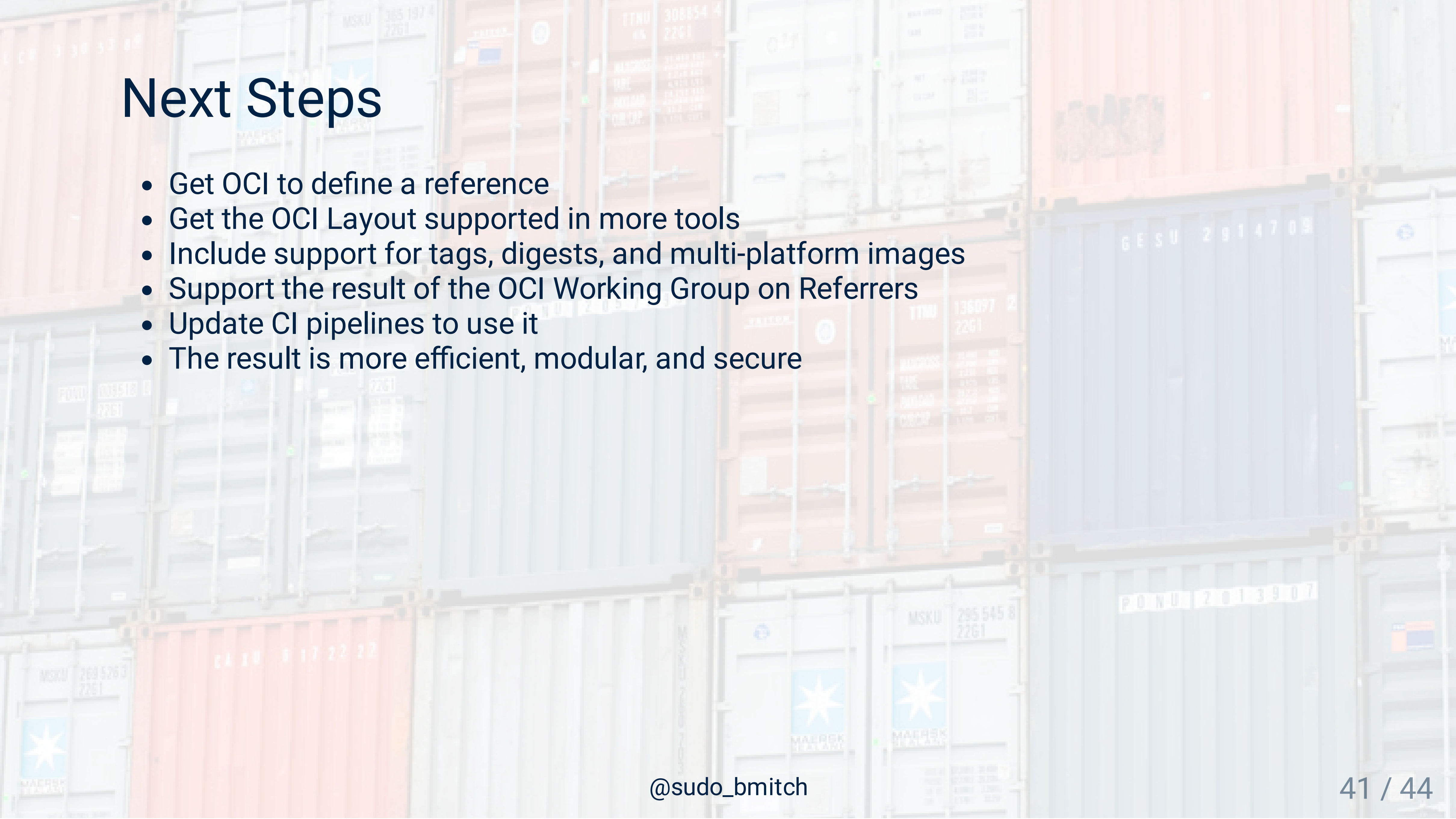
Sign

```
- name: zot
  run: |
    docker run --rm -d --name zot \
      -p 127.0.0.1:5000:5000 \
      -u "$(id -u):$(id -g)" \
      -v "$(pwd)/oci-layout:/var/lib/registry/demo" \
      ghcr.io/project-zot/zot-linux-amd64:v1.4.0
    curl -s http://localhost:5000/v2/ -o /dev/null || sleep 2

- name: sign
  env:
    COSIGN_EXPERIMENTAL: "true"
  run: |
    cosign sign "localhost:5000/demo:latest"
    cosign sign "localhost:5000/demo@${{ steps.artifacts.outputs.sbom_digest }}"
```

Push

```
- name: login and push
  run: |
    regctl registry login \
      -u "${{ secrets.GHCR_USER }}" \
      -p "${{ secrets.GHCR_TOKEN }}" \
      ghcr.io
    regctl image copy --digest-tags -v info \
      ocidir://oci-layout:latest \
      ghcr.io/sudo-bmitch/demo-gha-with-oci-layout:${{ steps.prep.outputs.version }}
```

Conclusion



Next Steps

- Get OCI to define a reference
- Get the OCI Layout supported in more tools
- Include support for tags, digests, and multi-platform images
- Support the result of the OCI Working Group on Referrers
- Update CI pipelines to use it
- The result is more efficient, modular, and secure

Open Source and Software Supply Chain Security

IBM is working with Open Source Security Foundation (OpenSSF) to help address challenges

IBM Point of View:

- ✓ Support open source communities security initiatives
- ✓ Learn about Software Bill of Materials (SBOM)
- ✓ Improved security execution is needed by all
- ✓ Contribute directly to open source projects you use

IBM commits to Cybersecurity training and secure software development

Find out more by reading our short blog here:
https://www.ibm.com/policy/open-source-security/

IBM **Developer**

Thank You

github.com/sudo-bmitch/presentations
github.com/sudo-bmitch/demo-gha-with-oci-layout

Brandon Mitchell
Twitter: @sudo_bmitch
GitHub: sudo-bmitch