one concern

Nothing is larger than life.™

Tech talk May 8th 2019 Datamon

Datamon manages infinite reflections of data

Background

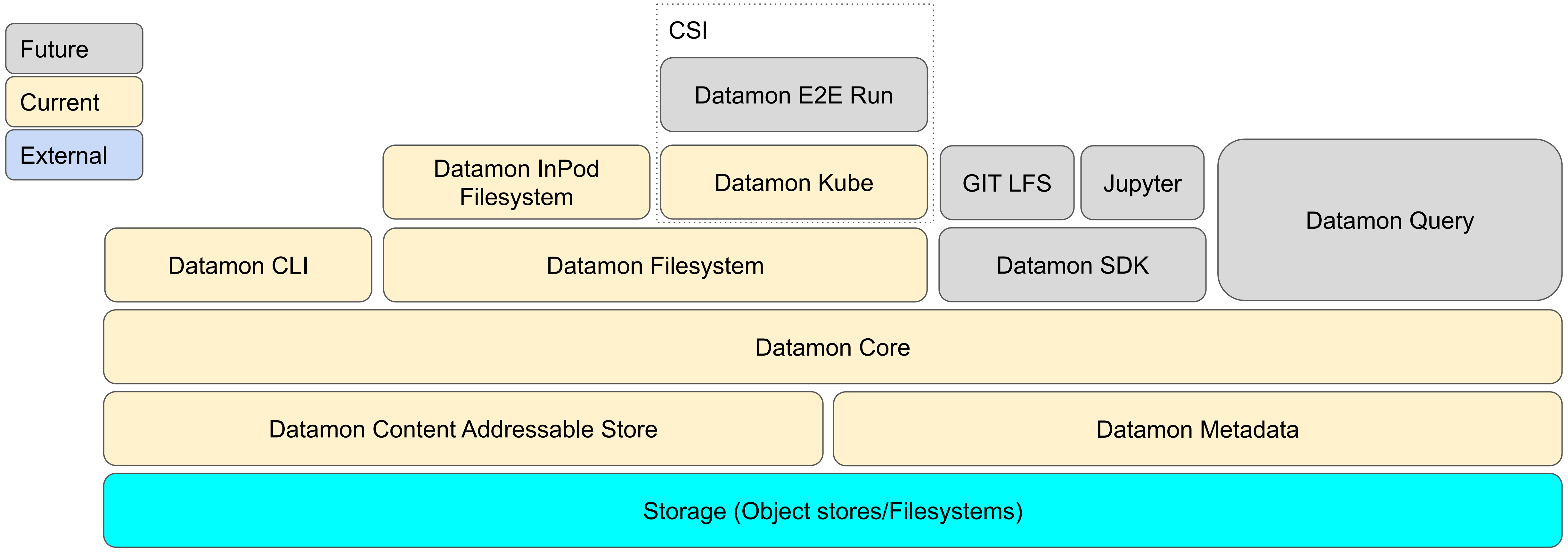
- Building a ML/AI app typically includes multiple stages that process data where the stage (compute) and the data have their own life cycles.
- The process to reverse engineering the output variations over time requires insight into every component and how they change over time.
- Insight into variations in components and how it impacts the end result are needed to automate promotion of individual components into production at scale.

Datamon

Datamon

Datamon: Git for data + more. Written entirely in GO

- Stores and versions all data under its management
- Understands how a ML pipeline is built and tracks versions of every component in the pipeline (compute + data)
- Can answer the question: What changed?
- Inexpensive and cheap
    - Inexpensive and easy to fork data and run experiments.
        - Deduplication of data
    - 10x cheaper than current solution
- Geo redundancy of data
    - Serverless functionality, geo replicated storage via GCS
- High concurrency of usage with “infinite” bandwidth.

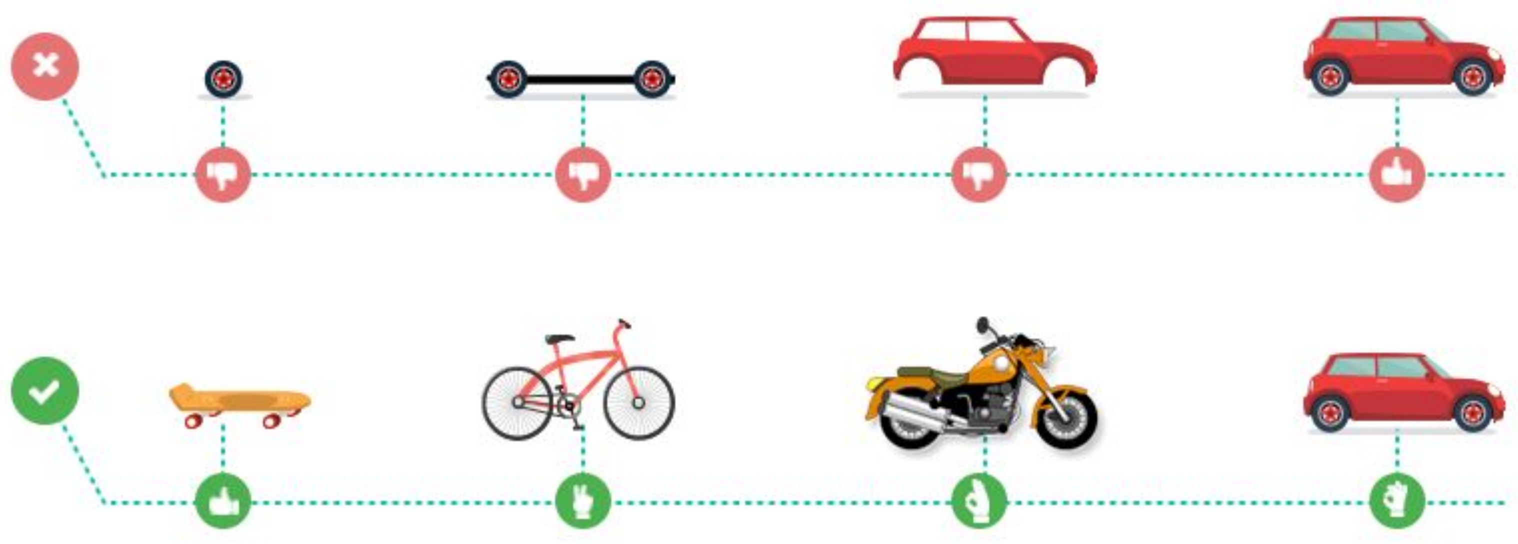


What does it mean?

- Treat it like git + dropbox
  - Any dataset that needs to be stored, create a personal repo and store it.
- Any python/fortran/go code that needs to process data in K8S use the sample yaml as template for fetching and storing data in Datamon
- On boarding
  - Try the CLI for personal use
  - Over the next few weeks existing workloads deployed in kubernetes will be rejuggled to fetch and store data in Datamon
- It is open source, tell your friends!

Development model

- Iterative
- Feedback is important
  - Feedback during on boarding will be used to guide the next set of features to work on. So provide feedback!
- You are the customer!
  - File issues!
  - Request features!



Source: https://www.quickscrum.com/Article/ArticleDetails/5174/1/Why-start-up-should-focus-on-Minimum-Viable-Product

Datamon Model

Repo: Repo is a unit of data that needs to exist together.

Bundle: Bundle analogous to git commit is a point in time copy/version of the data in a repo

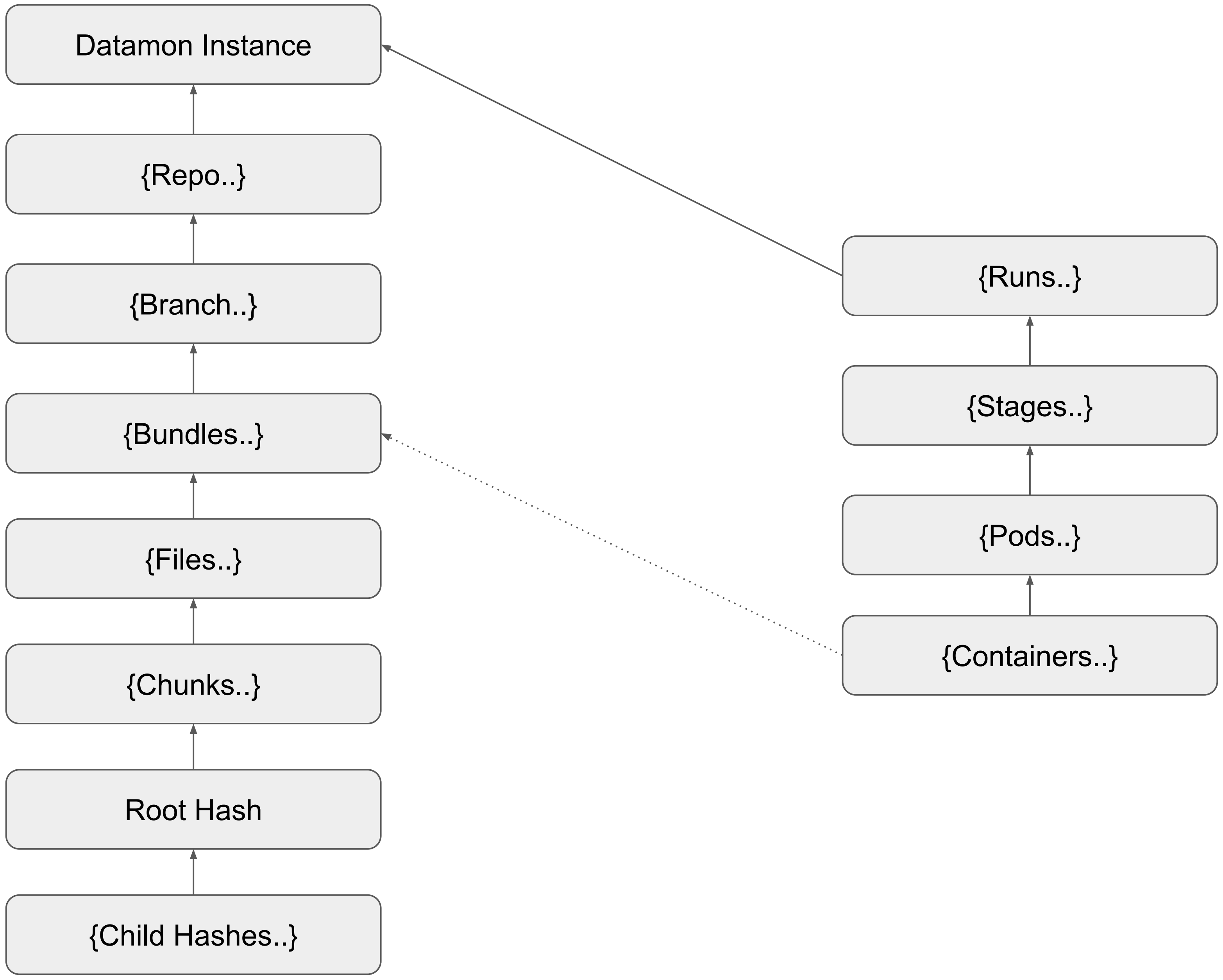
Planned features:

Tags: A name given to a bundle. Example: Latest, production

Branches: Different branches for the data in a repo.

Runs: ML pipeline run metadata that includes the versions of compute and data in use for a given run of a pipeline.

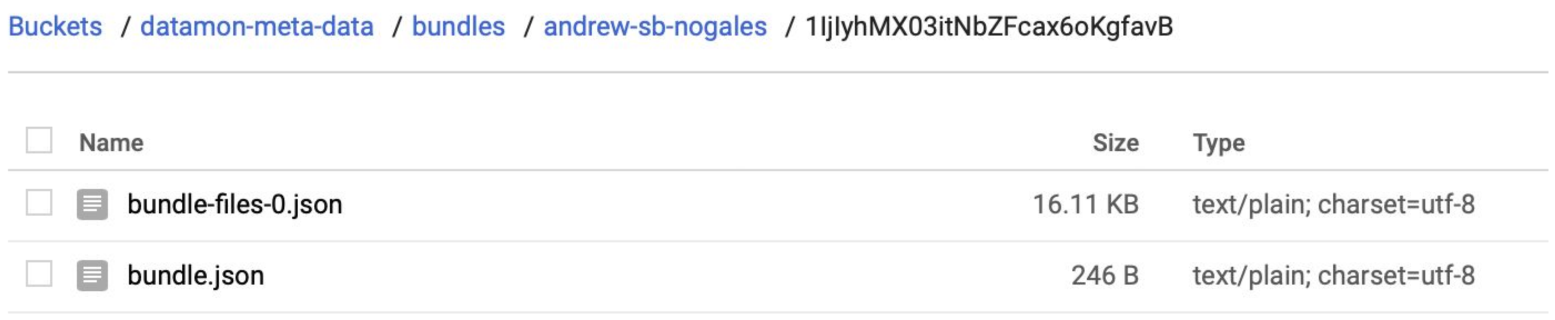
Datamon Core & Content Addressable Store



{..} Set of

Datamon Core

1. Allows for concurrent writes to the backend bucket without conflict
2. Chunks and hashes (blake) data to allow for deduplication at chunk level granularity (2 MB default)



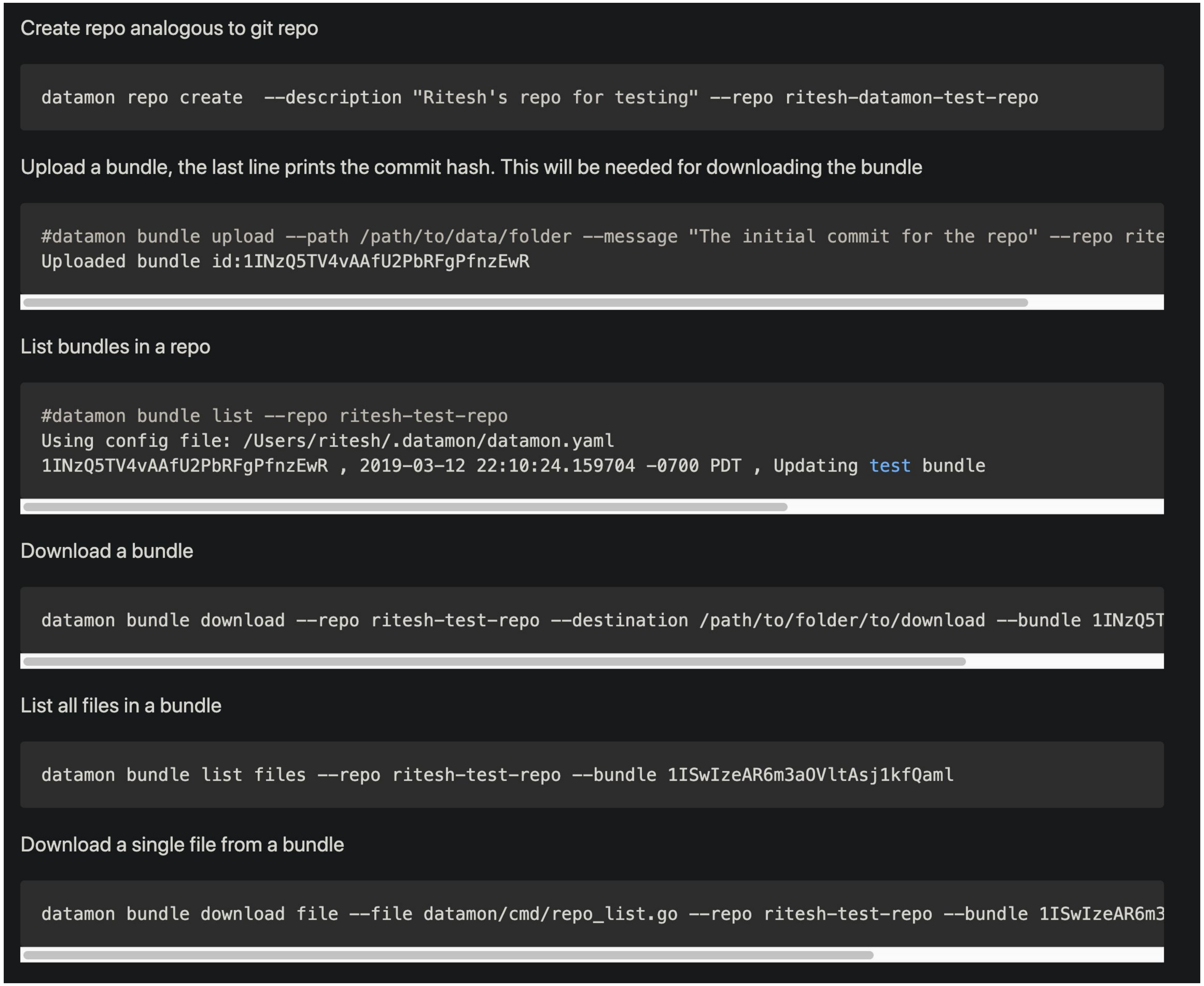
Buckets / datamon-meta-data / bundles / andrew-sb-nogales / 1IjIyhMX03itNbZFcax6oKgfavB

Name	Size	Type
bundle-files-0.json	16.11 KB	text/plain; charset=utf-8
bundle.json	246 B	text/plain; charset=utf-8

```
BundleEntries:
- hash: 4051e9747142924cf9a1d038f5eae458098dd1866ee8a93b991afa7b1e2d7cf3061f23530c26045d70e580403dcc17a88961ffe136d5303c1c06a98c9167174f
  name: SourceSink01.tim
  mode: 0
  size: 128997
- hash: 650ad11b5df4141095f9f304a235c1b0f7472bcc43fba40f9f788887319f6a30831b9800bc5197aedeaeaf7f741e335fcd324172c12a2c99f1adcba48608ba7d
  name: SourceSink02.tim
  mode: 0
  size: 132412
- hash: 1fe6abdae4da0d1ee2abd490c7b051a6be80b5cd6dfb0a9a79ae3e47d341188b91839c4891c77bf379768d256925796c33773f728c162a9d87fa9cb0dd7126ba
  name: SourceSink04.tim
  mode: 0
  size: 128580
- hash: cfda8edd42727e6c980d52438878d4deb5afbf41d270cd84c9ebc78918605241b681e4c5893bb248464b39ed1016f190de9cffe0d8b172956d99990b866a31a5
  name: SourceSink03.pli
  mode: 0
  size: 49
- hash: 6228085e7b369f1c12cd2f6079ad7f97e2184be368954ff8a6bd9c487388f816932d68981a42f71c3bd131647f5dce37ffe999419c95afa30e90b410b4706393
  name: SourceSink02.pli
  mode: 0
  size: 49
- hash: 8a3d4f5fe47ca5a291dd253876d4fcc287babc02c660e1a42edc2f393a1412f8c696cdee49543c170f8a51c52bdc57b8a2bab2ef57996fce7faa1108530565c1
  name: SourceSink05.pli
```

```
leafSize: 2097152
id: 1IjIyhMX03itNbZFcax6oKgfavB
message: Upload andrew-sb-nogales folder. Include new files from March 19th
timestamp: 2019-03-20T18:17:15.832041462Z
contributors:
- name: Ritesh H Shukla
  email: ritesh@oneconcern.com
count: 1
```

Datamon CLI



Create repo analogous to git repo

```
datamon repo create  --description "Ritesh's repo for testing" --repo ritesh-datamon-test-repo
```

Upload a bundle, the last line prints the commit hash. This will be needed for downloading the bundle

```
#datamon bundle upload --path /path/to/data/folder --message "The initial commit for the repo" --repo rite
Uploaded bundle id:1INzQ5TV4vAAfU2PbRFgPfnzEwR
```

List bundles in a repo

```
#datamon bundle list --repo ritesh-test-repo
Using config file: /Users/ritesh/.datamon/datamon.yaml
1INzQ5TV4vAAfU2PbRFgPfnzEwR , 2019-03-12 22:10:24.159704 -0700 PDT , Updating test bundle
```

Download a bundle

```
datamon bundle download --repo ritesh-test-repo --destination /path/to/folder/to/download --bundle 1INzQ5T
```

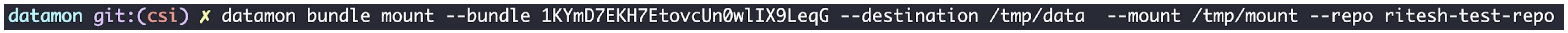
List all files in a bundle

```
datamon bundle list files --repo ritesh-test-repo --bundle 1ISwIzeAR6m3aOVltAsj1kfQaml
```

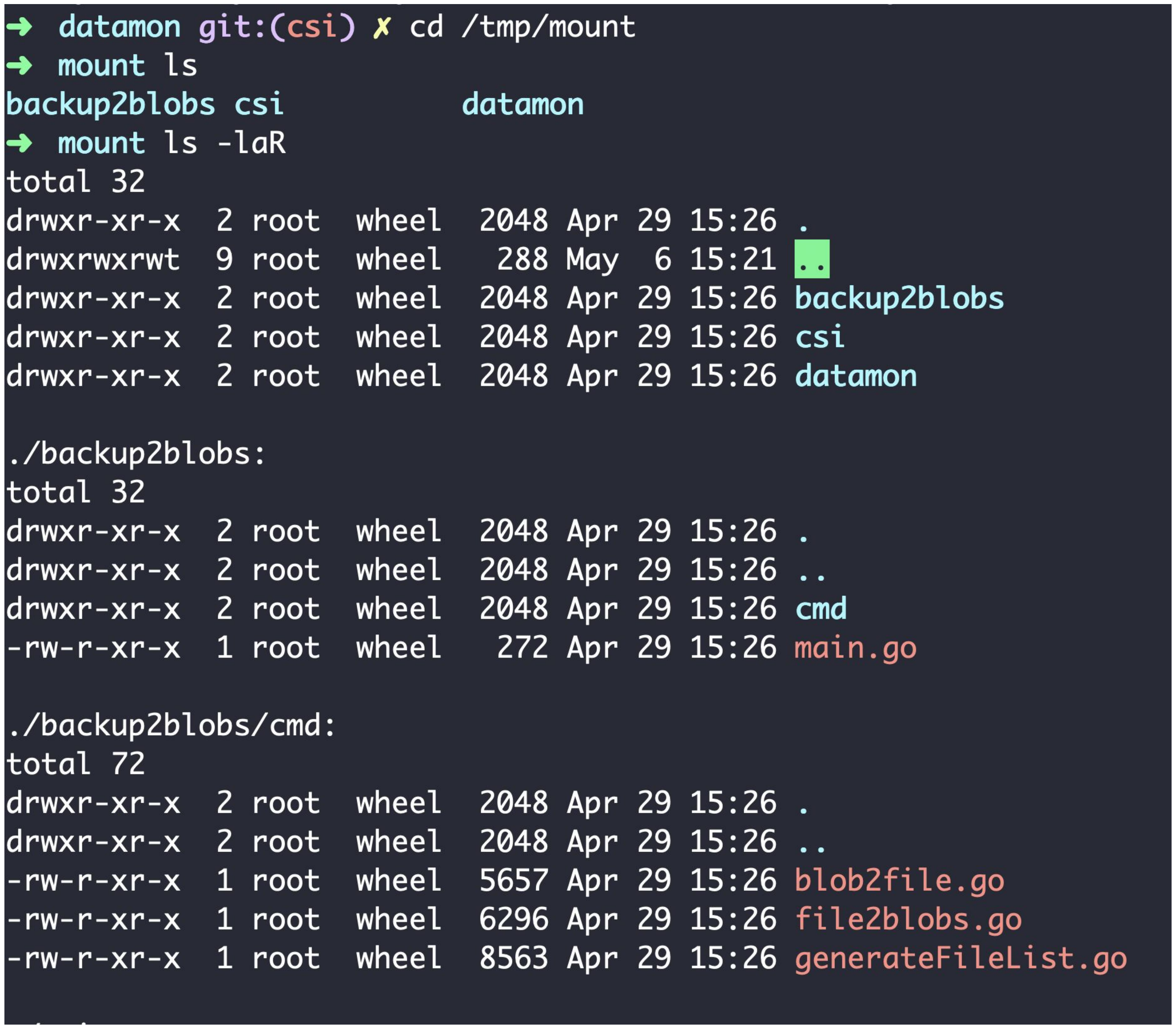
Download a single file from a bundle

```
datamon bundle download file --file datamon/cmd/repo_list.go --repo ritesh-test-repo --bundle 1ISwIzeAR6m3
```

Datamon Filesystem



```
datamon git:(csi) ✗ datamon bundle mount --bundle 1KYmD7EKH7EtovcUn0wlIX9LeqG --destination /tmp/data  --mount /tmp/mount --repo ritesh-test-repo
```



```
➜ datamon git:(csi) ✗ cd /tmp/mount
➜ mount ls
backup2blobs csi          datamon
➜ mount ls -laR
total 32
drwxr-xr-x  2 root  wheel  2048 Apr 29 15:26 .
drwxrwxrwt  9 root  wheel   288 May  6 15:21 ..
drwxr-xr-x  2 root  wheel  2048 Apr 29 15:26 backup2blobs
drwxr-xr-x  2 root  wheel  2048 Apr 29 15:26 csi
drwxr-xr-x  2 root  wheel  2048 Apr 29 15:26 datamon

./backup2blobs:
total 32
drwxr-xr-x  2 root  wheel  2048 Apr 29 15:26 .
drwxr-xr-x  2 root  wheel  2048 Apr 29 15:26 ..
drwxr-xr-x  2 root  wheel  2048 Apr 29 15:26 cmd
-rw-r-xr-x  1 root  wheel   272 Apr 29 15:26 main.go

./backup2blobs/cmd:
total 72
drwxr-xr-x  2 root  wheel  2048 Apr 29 15:26 .
drwxr-xr-x  2 root  wheel  2048 Apr 29 15:26 ..
-rw-r-xr-x  1 root  wheel  5657 Apr 29 15:26 blob2file.go
-rw-r-xr-x  1 root  wheel  6296 Apr 29 15:26 file2blobs.go
-rw-r-xr-x  1 root  wheel  8563 Apr 29 15:26 generateFileList.go
```

Kubernetes integration

1. Define a storage class that refers to the Datamon Repo

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: datamon
provisioner: com.datamon.csi
parameters:
 repo: "ritesh-test-repo"
```

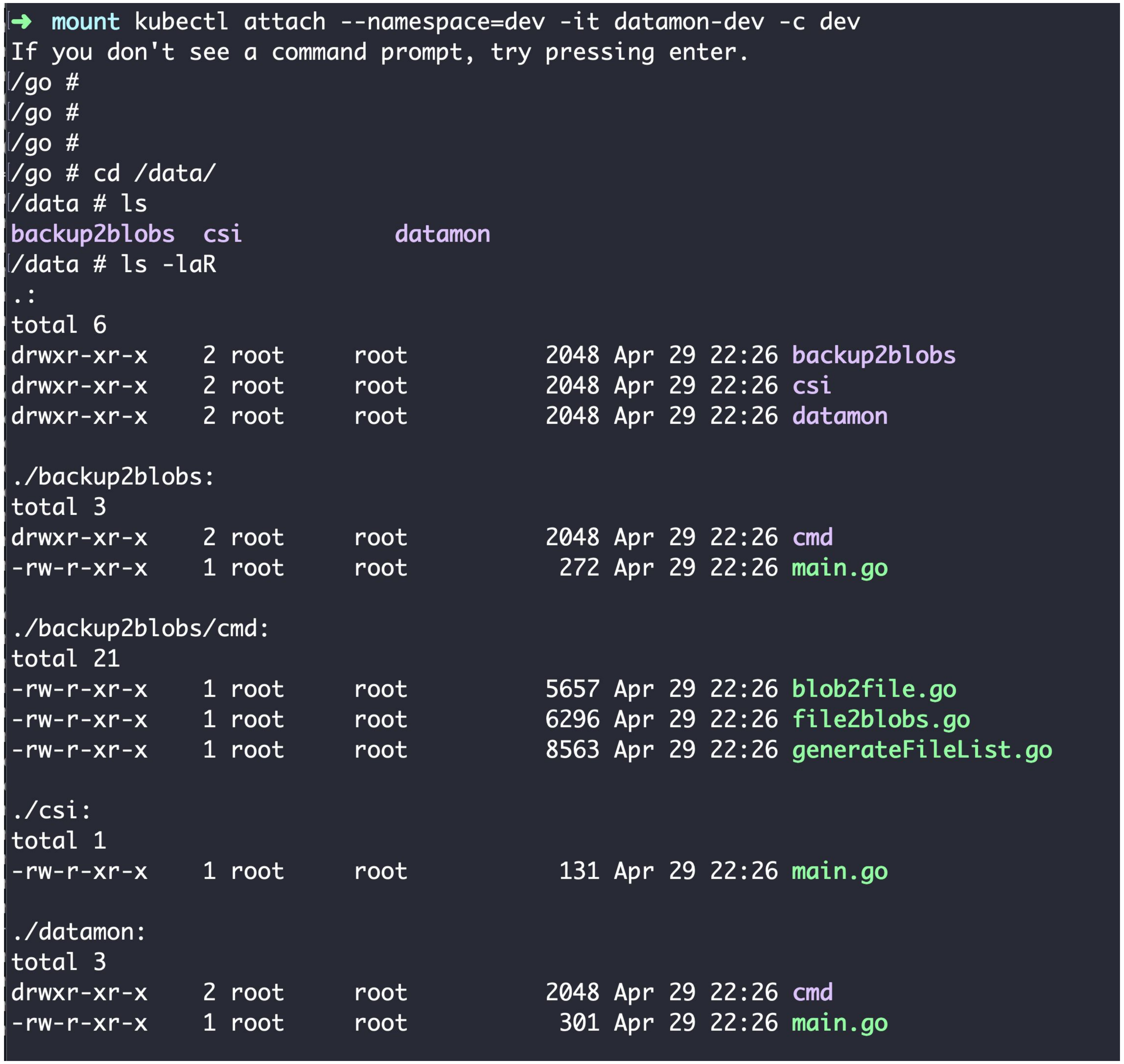
2. Define a claim that refers to the bundle

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: thankyoumsau42
spec:
 accessModes:
   - ReadOnlyMany
 storageClassName: datamon
 resources:
   requests:
     storage: 1Ti
 selector:
   matchLabels:
     release: "1KYmD7EKH7EtovcUn0wIIX9LeqG"
```

3. Define a pod that refers to the claim

```
apiVersion: v1
kind: Pod
metadata:
 # This name uniquely identifies the Deployment
 name: datamon-dev
spec:
 containers:
   - name: dev
     image: golang:1.11-alpine
     tty: true
     stdin: true
     volumeMounts:
       - name: credentials
         readOnly: true
         mountPath: "/etc/datamon-creds"
       - name: riteshpv2
         readOnly: true
         mountPath: "/data"
 volumes:
   - name: credentials
     secret:
       secretName: gcs-credentials
   - name: riteshpv2
     persistentVolumeClaim:
       claimName: thankyoumsau42
       readOnly: true
```

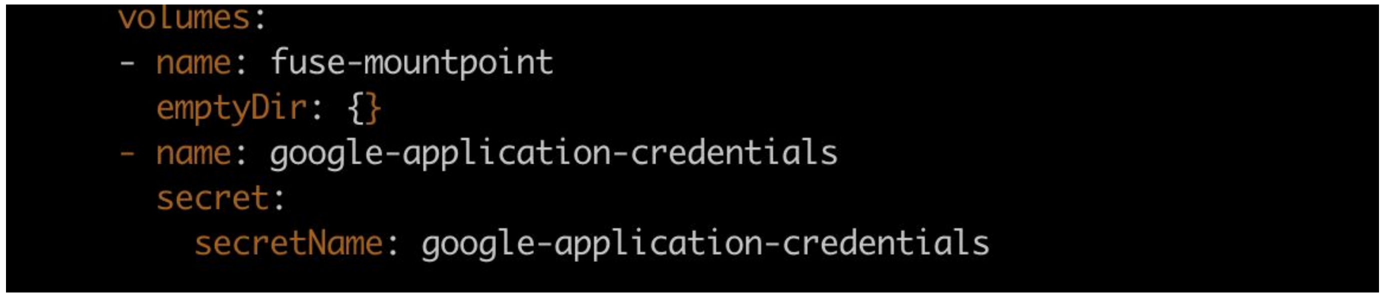
Kubernetes integration



```
➜  mount kubectl attach --namespace=dev -it datamon-dev -c dev
If you don't see a command prompt, try pressing enter.
/go #
/go #
/go #
/go # cd /data/
/data # ls
backup2blobs  csi           datamon
/data # ls -laR
.:
total 6
drwxr-xr-x    2 root     root          2048 Apr 29 22:26 backup2blobs
drwxr-xr-x    2 root     root          2048 Apr 29 22:26 csi
drwxr-xr-x    2 root     root          2048 Apr 29 22:26 datamon

./backup2blobs:
total 3
drwxr-xr-x    2 root     root          2048 Apr 29 22:26 cmd
-rw-r-xr-x    1 root     root           272 Apr 29 22:26 main.go

./backup2blobs/cmd:
total 21
-rw-r-xr-x    1 root     root          5657 Apr 29 22:26 blob2file.go
-rw-r-xr-x    1 root     root          6296 Apr 29 22:26 file2blobs.go
-rw-r-xr-x    1 root     root          8563 Apr 29 22:26 generateFileList.go

./csi:
total 1
-rw-r-xr-x    1 root     root           131 Apr 29 22:26 main.go

./datamon:
total 3
drwxr-xr-x    2 root     root          2048 Apr 29 22:26 cmd
-rw-r-xr-x    1 root     root           301 Apr 29 22:26 main.go
```

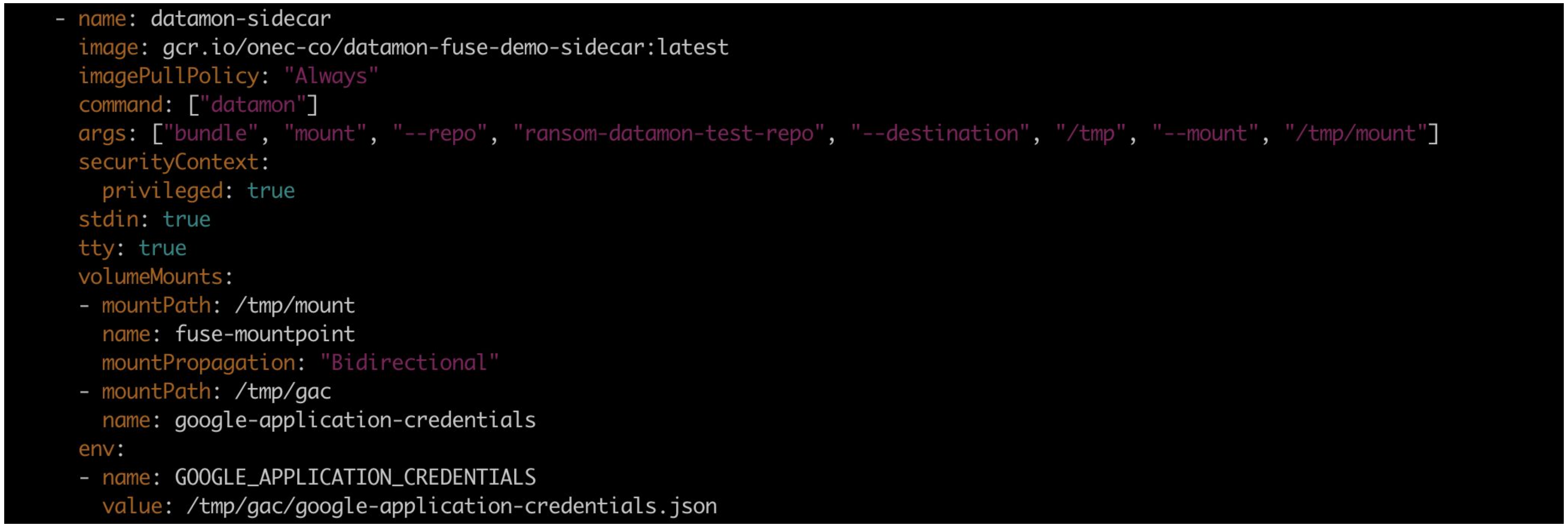
Datamon InPod Filesystem

1. Add a volume to contain mountpoints



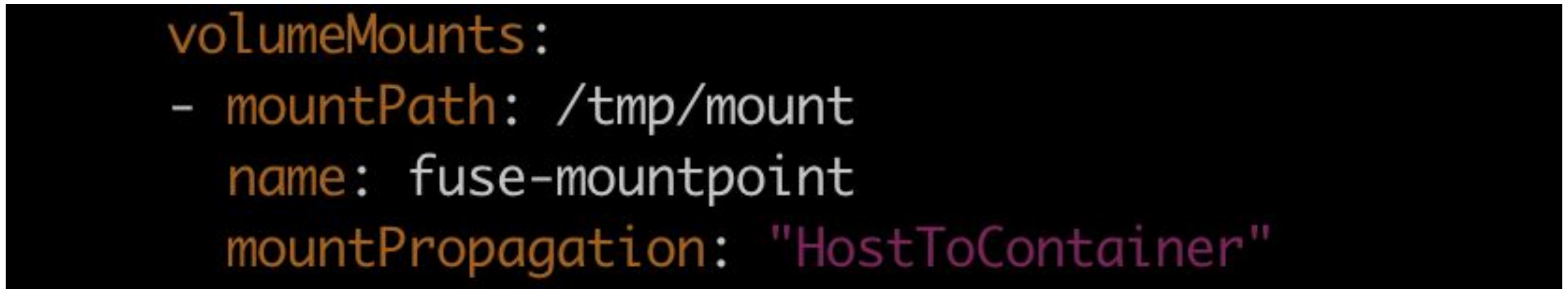
```yaml
volumes:
- name: fuse-mountpoint
  emptyDir: {}
- name: google-application-credentials
  secret:
    secretName: google-application-credentials
```

2. Add a InPod Filesystem container entry that provides bundles to the volume



```yaml
- name: datamon-sidecar
  image: gcr.io/onec-co/datamon-fuse-demo-sidecar:latest
  imagePullPolicy: "Always"
  command: ["datamon"]
  args: ["bundle", "mount", "--repo", "ransom-datamon-test-repo", "--destination", "/tmp", "--mount", "/tmp/mount"]
  securityContext:
    privileged: true
  stdin: true
  tty: true
  volumeMounts:
  - mountPath: /tmp/mount
    name: fuse-mountpoint
    mountPropagation: "Bidirectional"
  - mountPath: /tmp/gac
    name: google-application-credentials
  env:
  - name: GOOGLE_APPLICATION_CREDENTIALS
    value: /tmp/gac/google-application-credentials.json
```

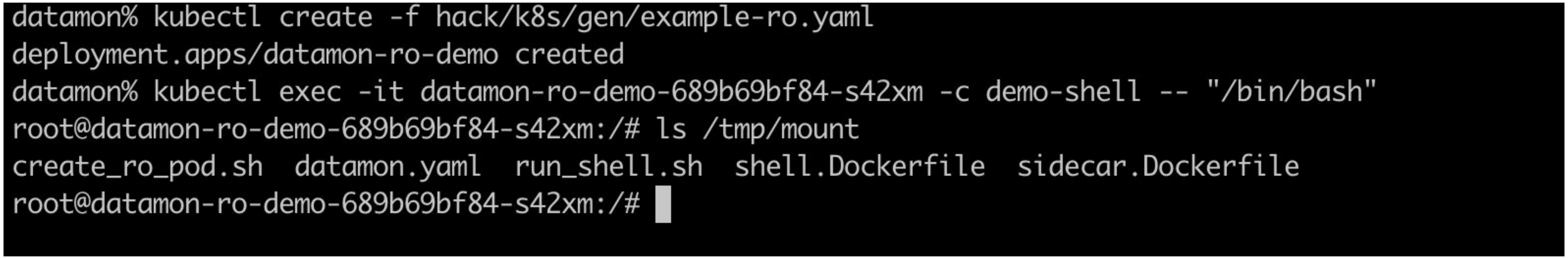
3. Add a volume entry to the application container



```yaml
volumeMounts:
- mountPath: /tmp/mount
  name: fuse-mountpoint
  mountPropagation: "HostToContainer"
```

Datamon InPod Filesystem

Then the bundle is available via filesystem operations in the application container



```
datamon% kubectl create -f hack/k8s/gen/example-ro.yaml
deployment.apps/datamon-ro-demo created
datamon% kubectl exec -it datamon-ro-demo-689b69bf84-s42xm -c demo-shell -- "/bin/bash"
root@datamon-ro-demo-689b69bf84-s42xm:/# ls /tmp/mount
create_ro_pod.sh  datamon.yaml  run_shell.sh  shell.Dockerfile  sidecar.Dockerfile
root@datamon-ro-demo-689b69bf84-s42xm:/#
```

Demo

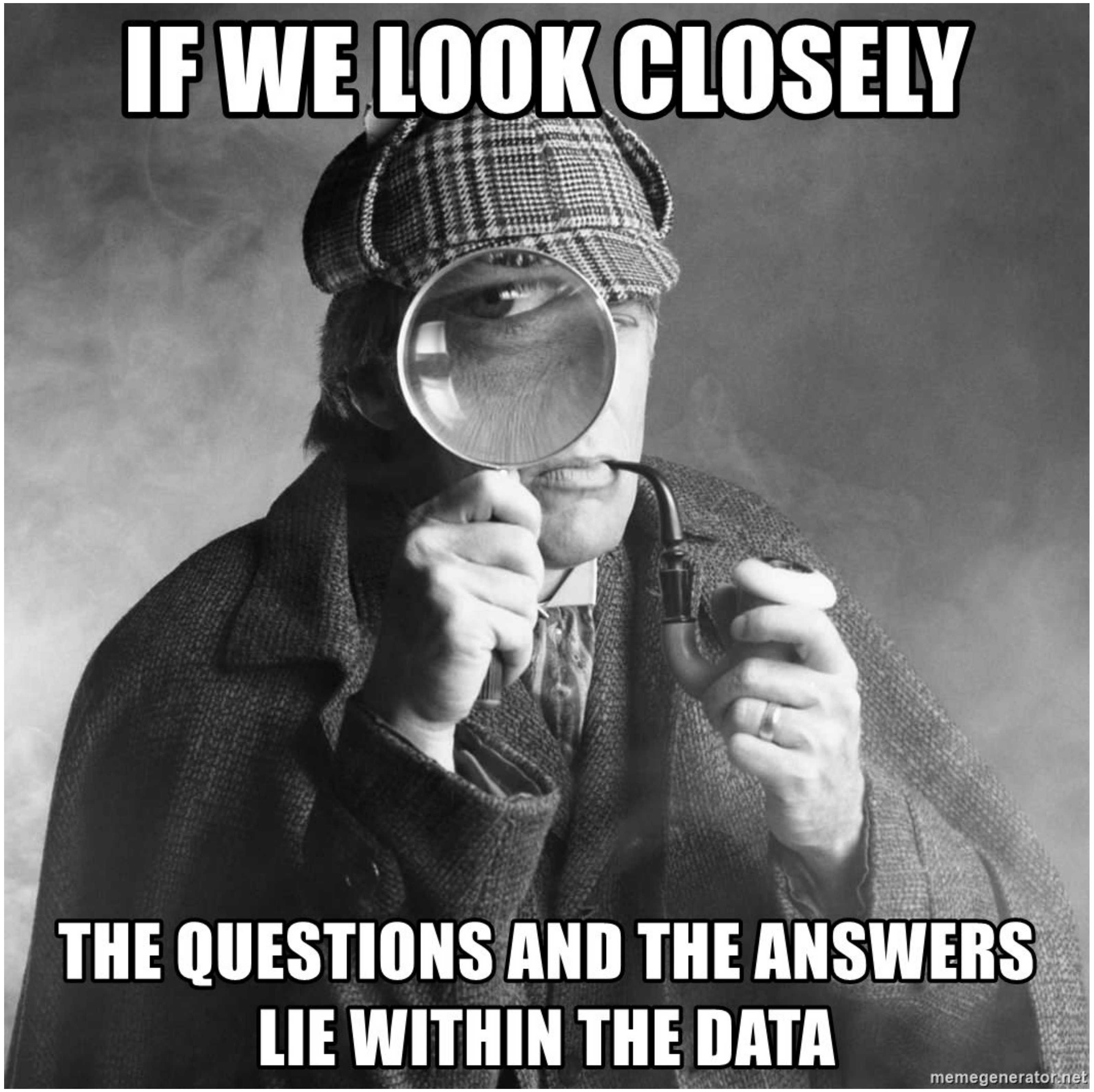
1. Kubernetes Volume
2. CLI

Next Steps: Integration & Query

1. Exiting pipelines moved to datamon
    a. Integrate
    b. Measure
    c. Improve
2. Track metadata for runs
    a. The metadata for repo/bundle/branch is linked for a pipeline
3. Track quality of output and allow experimentations at scale.
    a. Metadata allows for queries that span the pipeline to allow the following questions to be answered
        i. What changed between 2 runs?
        ii. If only one component is changed, how does the result vary?
4. Metadata tracking and query is a key value add beyond version management, efficiency and cost savings

What is next?

- New release coming next week
  - Kubernetes
  - InPod FS
  - More testing and bug fixes.
- Migration of pipelines:
  - Integration with existing pipelines
- Performance measurements and performance improvements
  - Where are the pain points that matter to OneConcern
- Argo integration
  - End to end visibility
- Datamon query
  - Query end to end visibility
  - Dive deep into changes over time
- Access control and encryption


IF WE LOOK CLOSELY
THE QUESTIONS AND THE ANSWERS LIE WITHIN THE DATA
memegenerator.net

Similar Projects?

- Pachyderm
- DVC.org
    - Deduplication of data at rest
    - Streaming and caching of data
    - Kubernetes integration
    - Tightly coupled with GIT