CERN

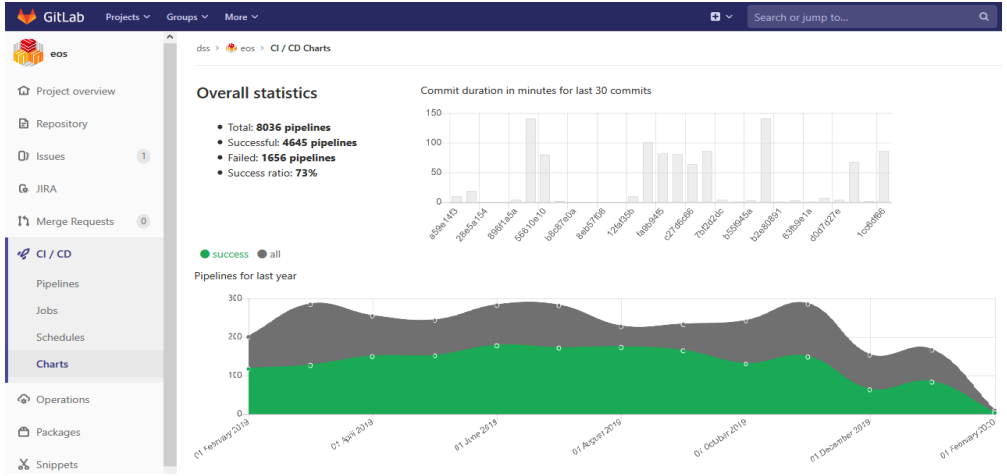
Testing EOS on Kubernetes

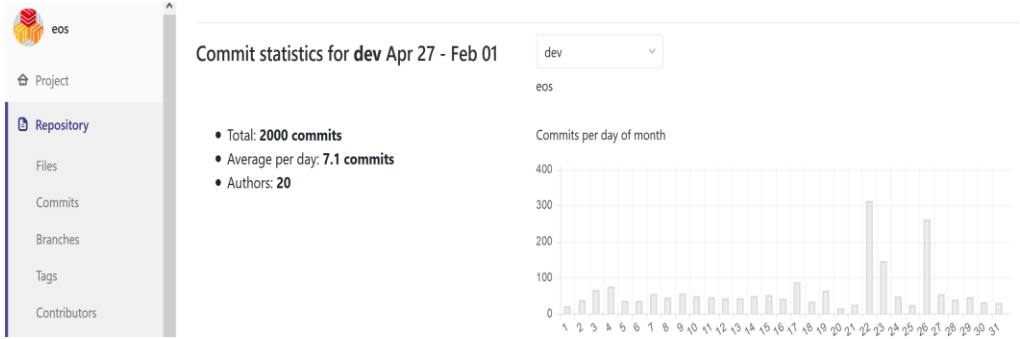
- Continuous Integration and beyond

Why testing

Why testing

- 600k lines of code, 1,5k files
- 340 PB of data (Physics, Home, Media, …)
- ~ dozen of devs, ops
- Thousands end users / customers

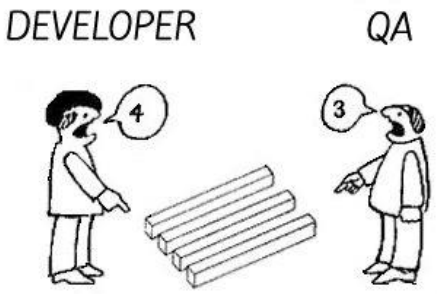




EOS Testing and Quality Assurance

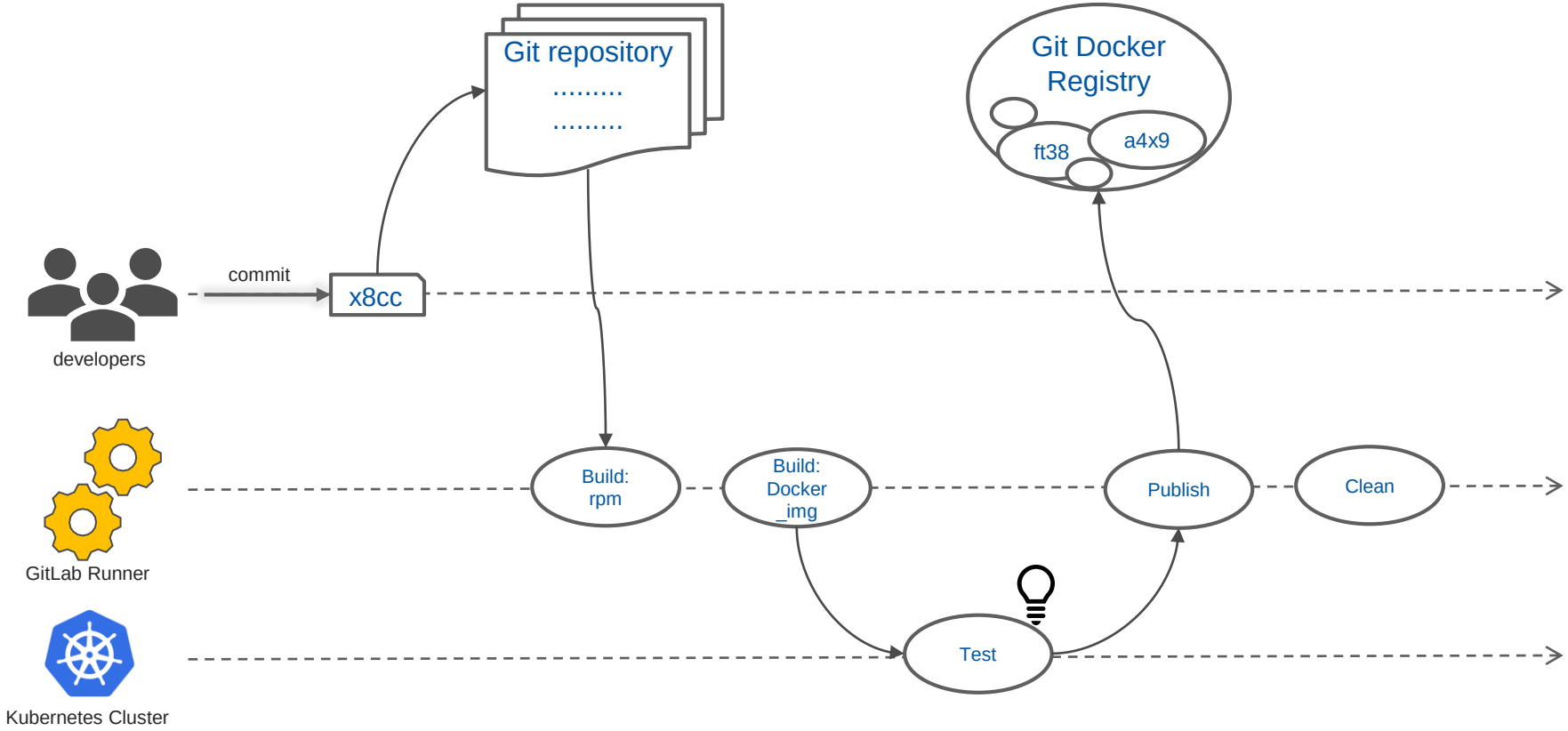
With such a large pool of parties involved, a testing framework is ~~desirable~~ **mandatory**, and must be:

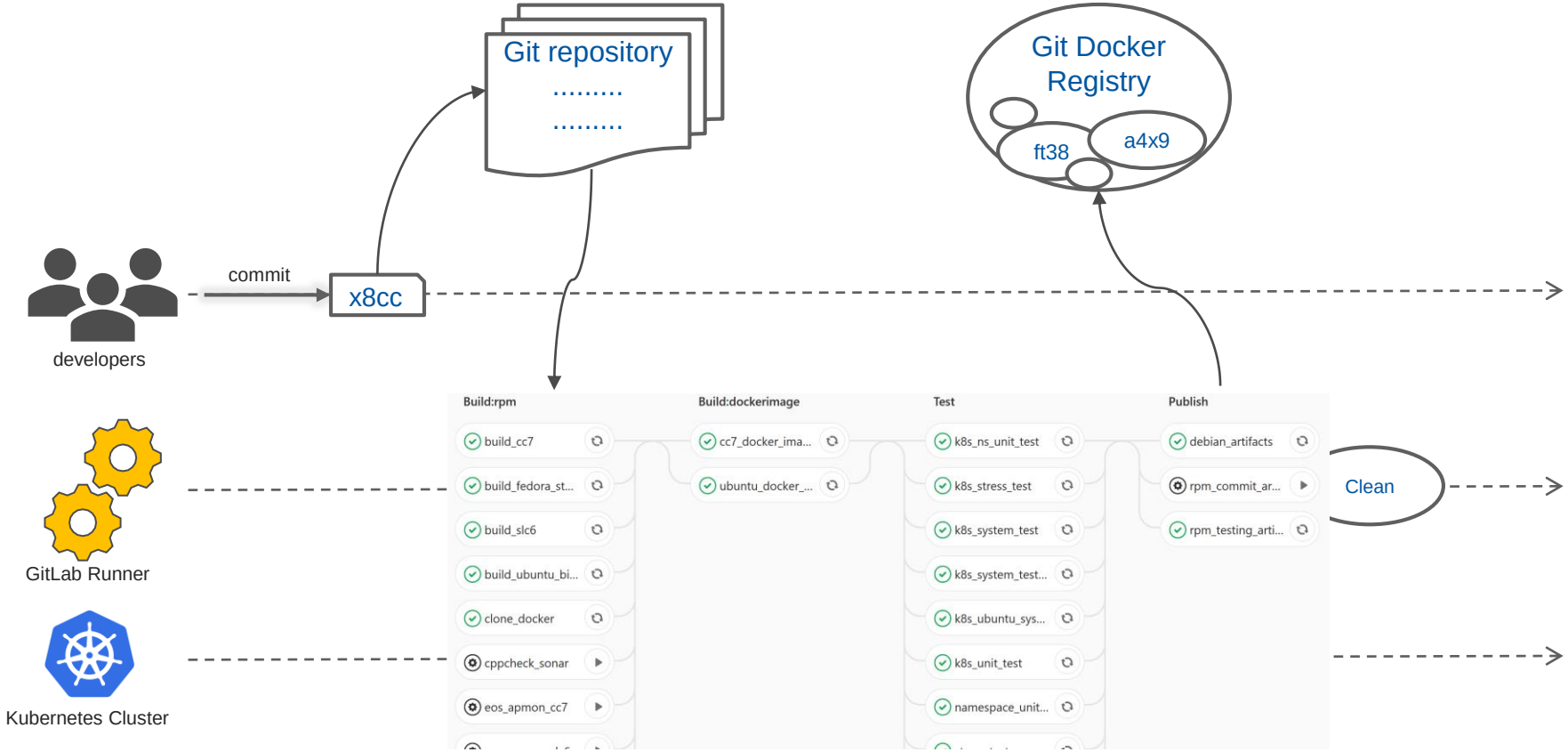
- Seamless to use
- Effective
- Efficient



How then?

How then?


Git repository
.....
.....
Git Docker Registry
ft38
a4x9
commit
x8cc
developers
GitLab Runner
Kubernetes Cluster
Build: rpm
Build: Docker _img
Test
Publish
Clean


Git repository
.....
.....
Git Docker Registry
ft38
a4x9
developers
commit
x8cc
GitLab Runner
Kubernetes Cluster
Build:rpm
build_cc7
build_fedora_st...
build_slc6
build_ubuntu_bi...
clone_docker
cppcheck_sonar
eos_apmon_cc7
Build:dockerimage
cc7_docker_ima...
ubuntu_docker_...
Test
k8s_ns_unit_test
k8s_stress_test
k8s_system_test
k8s_system_test...
k8s_ubuntu_sys...
k8s_unit_test
namespace_unit...
Publish
debian_artifacts
rpm_commit_ar...
rpm_testing_arti...
Clean

What it's left to be done

- Proof of concept K8s deployment
  - Template of independent instances (mgm, mq, qdb, kdc, fsts, clients,)
- Script that takes care of setup, config, and launch basic tests of "health state"
  - Existing tests must be plugged in
- Cluster can be ruled by CI Runner, but
  - Runner must be properly CI-labeled to automatically launch such tests
  - Add/modify stages in existing pipeline
  - Credentials for K8s private cluster must be fetched from secure storage, now injected
- More tests

(and counting …)

in the last episode…

Instantiating a test

1. Create a K8s Namespace
2. Create all K8s Services, Pods (1 or more per EOS role)
   - MGM, MQ, QDB, KRB5, FSTs, CLIENTs
3. Configure each Pod
   - “Patch” running containers to operate in the new network space
4. *Run the test you like most*

`create-all.sh` script

ref: https://gitlab.cern.ch/eos/eos-on-k8s

Instantiating a test

1. Create a K8s Namespace
2. Create all K8s Services, Pods (1 or more per EOS role)
   MGM, MQ, QDB, KRB5, FSTs, CLIENTs
3. Configure each Pod
   "Patch" running containers to operate in the new network space
4. *Run the test you like most*

```
$ create-all.sh  -n <k8s_namespace>
                 [-q] # with QuarkDB or not
                 [-m] <eos-mgm_count>
                 [-i] <eos_image_tag>
                 [-u] <eos_client_image_tag>
                 [-c] <eos-cli_count>
                 [-f] <eos-fst_count>
```

ref: https://gitlab.cern.ch/eos/eos-on-k8s

eos-mgm.template.yaml 2.72 KB

Edit | Web IDE | Replace | Delete

```yaml
### SERVICE ###
apiVersion: v1
kind: Service
metadata:
  name: &name eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    app: *name
  clusterIP: None # Need headless to resolve to Pod IP
  ports:
  - name: xrootd-mgm
    protocol: TCP
    port: 1094
    targetPort: 1094
  - name: xrootd-sync
    protocol: TCP
    port: 1096
    targetPort: 1096
  - name: xrootd-http
    protocol: TCP
    port: 8000
    targetPort: 8000
  - name: fusex
    protocol: TCP
    port: 1100
    targetPort: 1100

---

### DEPLOYMENT ###
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eos-mgm%%%NUMBER%%%-deployment
  labels:
    app: eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    matchLabels:
      app: eos-mgm%%%NUMBER%%%
  replicas: 1
  template:
    metadata:
      name: &name eos-mgm%%%NUMBER%%%
      labels:
        app: *name
    spec:
      hostname: *name
      subdomain: *name
      containers:
      - name: eos-storage
        image: gitlab-registry.cern.ch/dss/eos:%%%IMAGE_TAG%%%
        imagePullPolicy: IfNotPresent
        readinessProbe: # let it be startupProbe when deployed on kubectl >= 1.16
          exec:
            command:
            - /volumes/configmap/configure.sh
        command: ["/bin/sh", "-c"]
        args:
          - echo Starting::$(PODINFO_NAMESPACE).$(PODINFO_NAME);
            sleep infinity;
        env:
        ### PodInfo ###
        - name: PODINFO_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: PODINFO_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: PODINFO_IP
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
        - name: PODINFO_NODE_IP
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
        - name: PODINFO_NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        - name: EOS_QUARKDB_HOSTPORT
          value: "eos-qdb.eos-qdb.%%%NAMESPACE%%%.svc.cluster.local:7777"
        # - name: EOS_QUARKDB_PASSWD
        #   value: "password_must_be_at_least_32_characters"
        ### work around current Kubernetes setup issues. This should *NOT* be enabled in production
        - name: EOS_SKIP_SSS_HOSTNAME_MATCH
          value: "true"
        securityContext:
          privileged: true
          allowPrivilegeEscalation: true
          capabilities:
            add: ["SYS_PTRACE"]
        volumeMounts:
          - name: configmap-volume
            mountPath: /volumes/configmap/configure.sh
            subPath: configure.sh
      volumes:
      - name: configmap-volume
        configMap:
          name: configure-container
          defaultMode: 0755
```

eos-mgm.template.yaml 2.72 KB

Edit | Web IDE | Replace | Delete

```yaml
### SERVICE ###
apiVersion: v1
kind: Service
metadata:
  name: &name eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    app: *name
  clusterIP: None # Need headless to resolve to Pod IP
  ports:
  - name: xrootd-mgm
    protocol: TCP
    port: 1094
    targetPort: 1094
  - name: xrootd-sync
    protocol: TCP
    port: 1096
    targetPort: 1096
  - name: xrootd-http
    protocol: TCP
    port: 8000
    targetPort: 8000
  - name: fusex
    protocol: TCP
    port: 1100
    targetPort: 1100

---

### DEPLOYMENT ###
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eos-mgm%%%NUMBER%%%-deployment
  labels:
    app: eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    matchLabels:
      app: eos-mgm%%%NUMBER%%%
  replicas: 1
  template:
    metadata:
      name: &name eos-mgm%%%NUMBER%%%
      labels:
        app: *name
    spec:
      hostname: *name
      subdomain: *name
      containers:
      - name: eos-storage
        image: gitlab-registry.cern.ch/dss/eos:%%%IMAGE_TAG%%%
        imagePullPolicy: IfNotPresent
        readinessProbe: # let it be startupProbe when deployed on kubectl >= 1.16
          exec:
            command:
            - /volumes/configmap/configure.sh
        command: ["/bin/sh", "-c"]
        args:
          - echo Starting::$(PODINFO_NAMESPACE).$(PODINFO_NAME);
            sleep infinity;
        env:
        ### PodInfo ###
        - name: PODINFO_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: PODINFO_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: PODINFO_IP
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
        - name: PODINFO_NODE_IP
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
        - name: PODINFO_NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        - name: EOS_QUARKDB_HOSTPORT
          value: "eos-qdb.eos-qdb.%%%NAMESPACE%%%.svc.cluster.local:7777"
        # - name: EOS_QUARKDB_PASSWD
        #   value: "password_must_be_at_least_32_characters"
        ### work around current Kubernetes setup issues. This should *NOT* be enabled in production
        - name: EOS_SKIP_SSS_HOSTNAME_MATCH
          value: "true"
        securityContext:
          privileged: true
          allowPrivilegeEscalation: true
          capabilities:
            add: ["SYS_PTRACE"]
        volumeMounts:
          - name: configmap-volume
            mountPath: /volumes/configmap/configure.sh
            subPath: configure.sh
      volumes:
      - name: configmap-volume
        configMap:
          name: configure-container
          defaultMode: 0755
```

eos-mgm.template.yaml 2.72 KB

Edit | Web IDE | Replace | Delete

```yaml
### SERVICE ###
apiVersion: v1
kind: Service
metadata:
  name: &name eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    app: *name
  clusterIP: None # Need headless to resolve to Pod IP
  ports:
  - name: xrootd-mgm
    protocol: TCP
    port: 1094
    targetPort: 1094
  - name: xrootd-sync
    protocol: TCP
    port: 1096
    targetPort: 1096
  - name: xrootd-http
    protocol: TCP
    port: 8000
    targetPort: 8000
  - name: fusex
    protocol: TCP
    port: 1100
    targetPort: 1100

---

### DEPLOYMENT ###
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eos-mgm%%%NUMBER%%%-deployment
  labels:
    app: eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    matchLabels:
      app: eos-mgm%%%NUMBER%%%
  replicas: 1
  template:
    metadata:
      name: &name eos-mgm%%%NUMBER%%%
      labels:
        app: *name
    spec:
      hostname: *name
      subdomain: *name
      containers:
      - name: eos-storage
        image: gitlab-registry.cern.ch/dss/eos:%%%IMAGE_TAG%%%
        imagePullPolicy: IfNotPresent
        readinessProbe: # let it be startupProbe when deployed on kubectl >= 1.16
          exec:
            command:
            - /volumes/configmap/configure.sh
        command: ["/bin/sh", "-c"]
        args:
          - echo Starting::$(PODINFO_NAMESPACE).$(PODINFO_NAME);
            sleep infinity;
        env:
        ### PodInfo ###
        - name: PODINFO_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: PODINFO_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: PODINFO_IP
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
        - name: PODINFO_NODE_IP
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
        - name: PODINFO_NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        - name: EOS_QUARKDB_HOSTPORT
          value: "eos-qdb.eos-qdb.%%%NAMESPACE%%%.svc.cluster.local:7777"
        # - name: EOS_QUARKDB_PASSWD
        #   value: "password_must_be_at_least_32_characters"
        ### work around current Kubernetes setup issues. This should *NOT* be enabled in production
        - name: EOS_SKIP_SSS_HOSTNAME_MATCH
          value: "true"
        securityContext:
          privileged: true
          allowPrivilegeEscalation: true
          capabilities:
            add: ["SYS_PTRACE"]
        volumeMounts:
          - name: configmap-volume
            mountPath: /volumes/configmap/configure.sh
            subPath: configure.sh
      volumes:
      - name: configmap-volume
        configMap:
          name: configure-container
          defaultMode: 0755
```

pvc.template.yaml 624 Bytes

```yaml
### PERSISTENTVOLUMECLAIM ###
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: eos-allroles-varlogeos-pvc
  namespace: %%%NAMESPACE%%%
  labels:
    app: eos-allroles
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5G
  storageClassName: geneva-cephfs-testing
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: eos-%%%ROLE%%%%%%NUMBER%%%-volumesmnt-pvc
  namespace: %%%NAMESPACE%%%
  labels:
    app: eos-%%%ROLE%%%%%%NUMBER%%%
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10G
  storageClassName: geneva-cephfs-testing
```

pv-patch.template.yaml 1.1 KB

```yaml
spec:
  template:
    spec:
      initContainers:
      - name: varlogeos-volume-hack
        image: busybox
        command: ["/bin/sh", "-c"]
        args:
          - mkdir /var/log/eos/tx;
            touch /var/log/eos/tx/transfer-archive.log;
            chown -R 2:2 /var/log/eos;
            chmod 755 /var/log/eos /var/log/eos/tx;
            chmod 644 /var/log/eos/tx/transfer-archive.log;
        volumeMounts:
        - name: eos-allroles-varlogeos-pv
          mountPath: /var/log/eos
          subPath: eos
      containers:
      - name: eos-storage
        volumeMounts:
          - name: eos-allroles-varlogeos-pv
            mountPath: /var/log/eos
            subPath: eos
          - name: eos-%%%ROLE%%%%%%NUMBER%%%-volumesmnt-pv
            mountPath: /volumes/mnt
      volumes:
      - name: eos-allroles-varlogeos-pv
        persistentVolumeClaim:
          claimName: eos-allroles-varlogeos-pvc
          readOnly: false
      - name: eos-%%%ROLE%%%%%%NUMBER%%%-volumesmnt-pv
        persistentVolumeClaim:
          claimName: eos-%%%ROLE%%%%%%NUMBER%%%-volumesmnt-pvc
          readOnly: false
```

$ kubectl patch mydeployment mypatch-demo --patch "$(cat mypatch-file.yaml)"

eos-mgm.template.yaml 2.72 KB

```yaml
### SERVICE ###
apiVersion: v1
kind: Service
metadata:
  name: &name eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    app: *name
  clusterIP: None # Need headless to re
  ports:
  - name: xrootd-mgm
    protocol: TCP
    port: 1094
    targetPort: 1094
  - name: xrootd-sync
    protocol: TCP
    port: 1096
    targetPort: 1096
  - name: xrootd-http
    protocol: TCP
    port: 8000
    targetPort: 8000
  - name: fusex
    protocol: TCP
    port: 1100
    targetPort: 1100

---

### DEPLOYMENT ###
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eos-mgm%%%NUMBER%%%-deployment
  labels:
    app: eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    matchLabels:
      app: eos-mgm%%%NUMBER%%%
  replicas: 1
  template:
    metadata:
      name: &name eos-mgm%%%NUMBER%%%
      labels:
        app: *name
    spec:
      hostname: *name
      subdomain: *name
      containers:
```

eos-fst.template.yaml 2.37 KB

Edit | Web IDE | Replace | Delete

```yaml
### SERVICE ###
apiVersion: v1
kind: Service
metadata:
  name: &name eos-fst%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    app: *name
  clusterIP: None # Need headless to resolve to Pod IP
  ports:
  - name: xrootd-fst
    protocol: TCP
    port: 1095
    targetPort: 1095
  - name: xrootd-8001
    protocol: TCP
    port: 8001
    targetPort: 8001

---

### DEPLOYMENT ###
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eos-fst%%%NUMBER%%%-deployment
  labels:
    app: eos-fst%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    matchLabels:
      app: eos-fst%%%NUMBER%%%
  replicas: 1
  template:
    metadata:
      name: &name eos-fst%%%NUMBER%%%
      labels:
        app: *name
    spec:
      hostname: *name
      subdomain: *name
      containers:
      - name: eos-storage
        image: gitlab-registry.cern.ch/dss/eos:%%%IMAGE_TAG%%%
        imagePullPolicy: IfNotPresent
        readinessProbe: # let it be startupProbe when deployed on kubectl >= 1.16
          exec:
            command:
            - /volumes/configmap/configure.sh
        command: ["/bin/sh", "-c"]
        args:
          - echo Starting::$(PODINFO_NAMESPACE).$(PODINFO_NAME);
            sleep infinity;
        env:
        ### PodInfo ###
        - name: PODINFO_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: PODINFO_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: PODINFO_IP
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
        - name: PODINFO_NODE_IP
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
        - name: PODINFO_NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        ### work around current Kubernetes setup issues. This should *NOT* be enabled in production
        - name: EOS_SKIP_SSS_HOSTNAME_MATCH
          value: "true"
        securityContext:
          privileged: true
          allowPrivilegeEscalation: true
          capabilities:
            add: ["SYS_PTRACE"]
        volumeMounts:
          - name: configmap-volume
            mountPath: /volumes/configmap/configure.sh
            subPath: configure.sh
      volumes:
      - name: configmap-volume
        configMap:
          name: configure-container
          defaultMode: 0755
```

eos-mgm.template.yaml 2.72 KB

```yaml
### SERVICE ###
apiVersion: v1
kind: Service
metadata:
  name: &name eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    app: *name
  clusterIP: None # Need headless to resolve to Pod IP
  ports:
  - name: xrootd-mgm
    protocol: TCP
    port: 1094
    targetPort: 1094
  - name: xrootd-sync
    protocol: TCP
    port: 1096
    targetPort: 1096
  - name: xrootd-http
    protocol: TCP
    port: 8000
    targetPort: 8000
  - name: fusex
    protocol: TCP
    port: 1100
    targetPort: 1100

---

### DEPLOYMENT ###
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eos-mgm%%%NUMBER%%%-deployment
  labels:
    app: eos-mgm%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    matchLabels:
      app: eos-mgm%%%NUMBER%%%
  replicas: 1
  template:
    metadata:
      name: &name eos-mgm%%%NUMBER%%%
      labels:
        app: *name
    spec:
      hostname: *name
      subdomain: *name
      containers:
```

eos-fst.template.yaml 2.37 KB

```yaml
### SERVICE ###
apiVersion: v1
kind: Service
metadata:
  name: &name eos-fst%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    app: *name
  clusterIP: None # Need headless to resolve to Pod IP
  ports:
  - name: xrootd-fst
    protocol: TCP
    port: 1095
    targetPort: 1095
  - name: xrootd-8001
    protocol: TCP
    port: 8001
    targetPort: 8001

---

### DEPLOYMENT ###
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eos-fst%%%NUMBER%%%-deployment
  labels:
    app: eos-fst%%%NUMBER%%%
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    matchLabels:
      app: eos-fst%%%NUMBER%%%
  replicas: 1
  template:
    metadata:
      name: &name eos-fst%%%NUMBER%%%
      labels:
        app: *name
    spec:
      hostname: *name
      subdomain: *name
```

eos-qdb.template.yaml 2.28 KB

Edit | Web IDE | Replace | Delete

```yaml
### SERVICE ###
apiVersion: v1
kind: Service
metadata:
  name: &name eos-qdb
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    app: *name
  clusterIP: None # Need headless to resolve to Pod IP
  ports: # @todo check the port to open/expose
  - name: xrootd-http
    protocol: TCP
    port: 7777
    targetPort: 7777

---

### DEPLOYMENT ###
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eos-qdb-deployment
  labels:
    app: eos-qdb
  namespace: %%%NAMESPACE%%%
spec:
  selector:
    matchLabels:
      app: eos-qdb
  replicas: 1
  template:
    metadata:
      name: &name eos-qdb
      labels:
        app: *name
    spec:
      hostname: *name
      subdomain: *name
      containers:
      - name: eos-storage
        image: gitlab-registry.cern.ch/dss/eos:%%%IMAGE_TAG%%%
        imagePullPolicy: IfNotPresent
        readinessProbe: # let it be startupProbe when deployed on kubectl >= 1.16
          exec:
            command:
            - /volumes/configmap/configure.sh
        command: ["/bin/sh", "-c"]
        args:
          - echo Starting::$(PODINFO_NAMESPACE).$(PODINFO_NAME);
            sleep infinity;
        env:
        ### PodInfo ###
        - name: PODINFO_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: PODINFO_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: PODINFO_IP
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
        - name: PODINFO_NODE_IP
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
        - name: PODINFO_NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        ### work around current Kubernetes setup issues. This should *NOT* be enabled in production
        - name: EOS_SKIP_SSS_HOSTNAME_MATCH
          value: "true"
        securityContext:
          privileged: true
          allowPrivilegeEscalation: true
          capabilities:
            add: ["SYS_PTRACE"]
        volumeMounts:
          - name: configmap-volume
            mountPath: /volumes/configmap/configure.sh
            subPath: configure.sh
      volumes:
      - name: configmap-volume
        configMap:
          name: configure-container
          defaultMode: 0755
```

Tests you can run (we can run)

Almost everything (in parallel too!) e.g. :

- Compact (MGM and QDB on the same machine) vs Frontend (MGM and QDB on different machines) deployment
- Node/Service failure (MGM master/slave behavior)
- Different scenarios stress tests (many clients, specific configs…)
- Existing tests now living out of GitlabCI
- Mixed release deployment (legacy clients? QDB?)

Tests you can run (we can run)

Almost everything (in parallel too!)

- Compact (MGM and QDB on the same machine) vs Frontend (MGM and QDB on different machines) deployment
- Node/Service failure (MGM master/slave behavior)
- Different scenarios stress tests (many clients, specific configs…)
- Existing tests now living out of GitlabCI
- Mixed release deployment (legacy clients? QDB?)

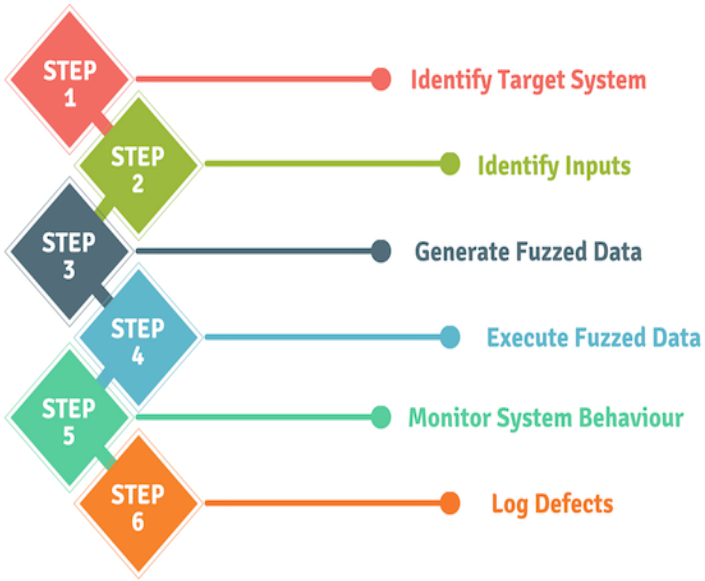
- Plug & Play, as long as they run on a standard EOS instance
- Need to check "Plug&Play-ability" for custom cases
(e.g. big cluster and k8s-native resource mgmt for wearing runs)

… beyond ?

defects bugs
software security attack
fail segmentation fault
overflows
black box testing fault injection vulnerabilities
software testing buffer
random data

fuzzing

eos-fuzz[1]



Two types of fuzz testing techniques

1. Naïve fuzz testing:
   - Input changes randomly

2. Structure-aware fuzz testing:
   - Input changes leveraging knowledge of data format

[1.] (private branch)

“dumb” fuzzer

```
10  extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {
11
12      std::string input {reinterpret_cast<char*>(const_cast<uint8_t*>(data))};
13      input.insert(0, "eos.rgid=0&eos.ruid=0&mgm.cmd.proto=");
14
15      // XrdMgmOfs* gofs is the global handle to a XrdMgmOfs object
16      auto file = gofs->newFile(); //  XrdSfsFile* file;
17      auto client = new XrdSecEntity(); //  XrdSecEntity* client;
18
19      file->open(path, 0, 0, client, input.c_str());
20
21      delete file;
22      delete client;
23
24      return 0;
25  }
26
```

custom fuzzer

```
10  extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {
11
12      std::string input {reinterpret_cast<char*>(const_cast<uint8_t*>(data))};
13      input.insert(0, "eos.rgid=0&eos.ruid=0&mgm.cmd.proto=");
14
15      // XrdMgmOfs* gofs is the global handle to a XrdMgmOfs object
16      auto file = gofs->newFile(); //  XrdSfsFile* file;
17      auto client = new XrdSecEntity(); //  XrdSecEntity* client;
18
19      file->open(path, 0, 0, client, input.c_str());
20
21      delete file;
22      delete client;
23
24      return 0;
25  }
```

```
26
27  #ifdef CUSTOM_MUTATOR
28
29  extern "C" size_t LLVMFuzzerCustomMutator(uint8_t *data, size_t size, size_t MaxSize, unsigned int Seed) {
30      // ...
31      return LLVMFuzzerMutate(data, size, size_t MaxSize);
32
33  }
34
35  #endif // CUSTOM_MUTATOR
```

“smart” structure-aware (protobuff) fuzzer

```
12  #include "libprotobuf-mutator/src/mutator.h"
13  #include "libprotobuf-mutator/src/mutator.cc"
14
15  #include "libprotobuf-mutator/src/libfuzzer/libfuzzer_macro.h"
16
17
18  extern "C" int LLVMFuzzerInitialize(int *argc, char ***argv) { ...
91  }
92
93  /* and here the magic happens */
94  DEFINE_PROTO_FUZZER(const eos::console::RequestProto& message) {
95
96    std::string b64buff;
97    eos::common::SymKey::ProtobufBase64Encode(&message, b64buff);
98
99    // ...
100   file->open(path, 0, 0, client, input.c_str());
101
102   delete file;
103   delete sec_ent;
104
105 }
```

```
$ clang -g -fsanitize=fuzzer,address -DCUSTOM_MUTATOR protofuzzer.cc

$ ./a.out

INFO: Seed: 3918206239
INFO: Loaded 1 modules (14 guards): [0x73be00, 0x73be38),
INFO: Loaded 1 PC tables (7 PCs): 7 [0x52f8c8,0x52f938),
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: A corpus is not provided, starting from an empty corpus
#0      READ units: 1
#1      INITED cov: 3 ft: 3 corp: 1/1b exec/s: 0 rss: 26Mb
#8      NEW    cov: 4 ft: 4 corp: 2/29b exec/s: 0 rss: 26Mb L: 28 MS: 2 InsertByte-InsertRepeatedBytes-
#3405   NEW    cov: 5 ft: 5 corp: 3/82b exec/s: 0 rss: 27Mb L: 53 MS: 4 InsertByte-EraseBytes-...
#8664   NEW    cov: 6 ft: 6 corp: 4/141b exec/s: 0 rss: 27Mb L: 59 MS: 3 CrossOver-EraseBytes-...
#272167 NEW    cov: 7 ft: 7 corp: 5/201b exec/s: 0 rss: 51Mb L: 60 MS: 1 InsertByte-
=================================================================
==2335==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000155c13 at pc 0x0000004ee637...
READ of size 1 at 0x602000155c13 thread T0
```

refs:
https://llvm.org/docs/LibFuzzer.html
https://github.com/google/libprotobuf-mutator

Does it work?

Does it work?

JIRA Dashboards Projects Issues Boards WBS Gantt-Chart CERNforge Create

New search

Find filters

FILTERS

My open issues

Reported by me

All issues

Open issues

Done issues

Viewed recently

Created recently

Resolved recently

Updated recently

FAVORITE FILTERS

You don't have any favorite filters.

Project: All Type: All Status: All Assignee: All Contains text More Search Advanced

Reporter: Current User

Order by Created

EOS-3904
'fs boot' command suffers from integer...

EOS-3903
'fs rm' comman throws std::out_of_rang...

EOS-3902
'group ls' command suffers from intege...

EOS-3870
command "eos rm ****" make the MG...

EOS / EOS-3870

command "eos rm ****" make the MGM crash

Comment Agile Board More Reopen Issue

Details

Type:	Bug	Status:	CLOSED (View Workflow)
Priority:	Minor		
Affects Version/s:	None	Resolution:	Fixed
Component/s:	None	Fix Version/s:	None
		Security Level:	Internal Data (Only authenticated CERN users can see this issue)
Labels:	None		

Description

Using continuous fuzz tests against a few "personal" eos instances, I found that the command <eos rm ****> eventually causes the crash of the MGM (note, apparently from the 4th star, but not with less).

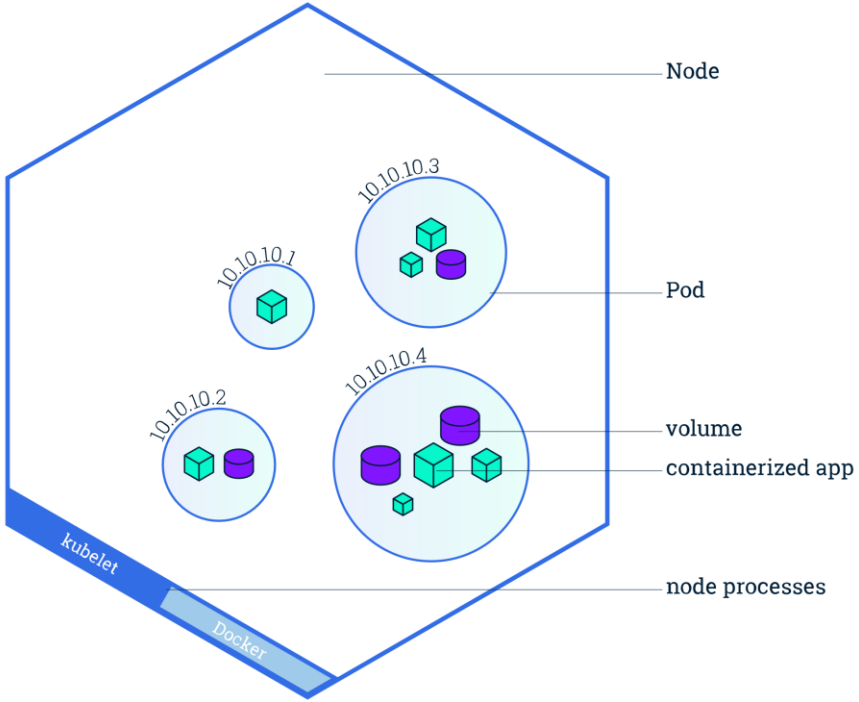
"That's all Folks!"

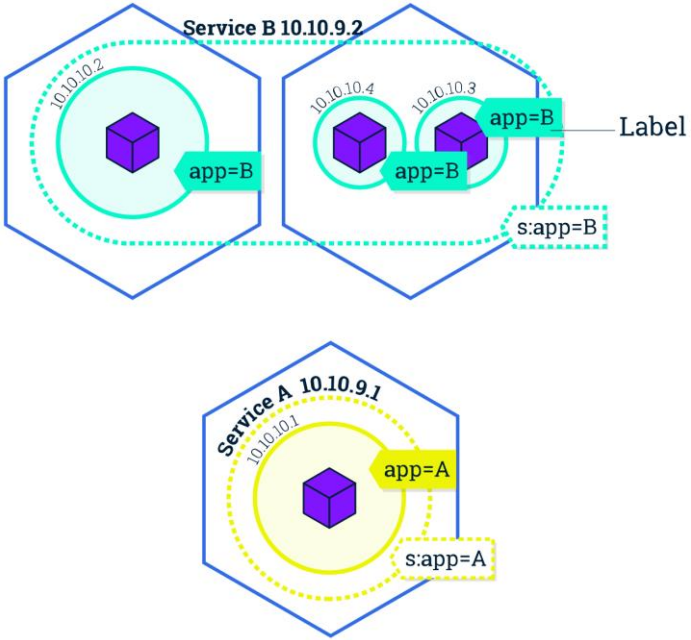
Thanks for your attention

CERN

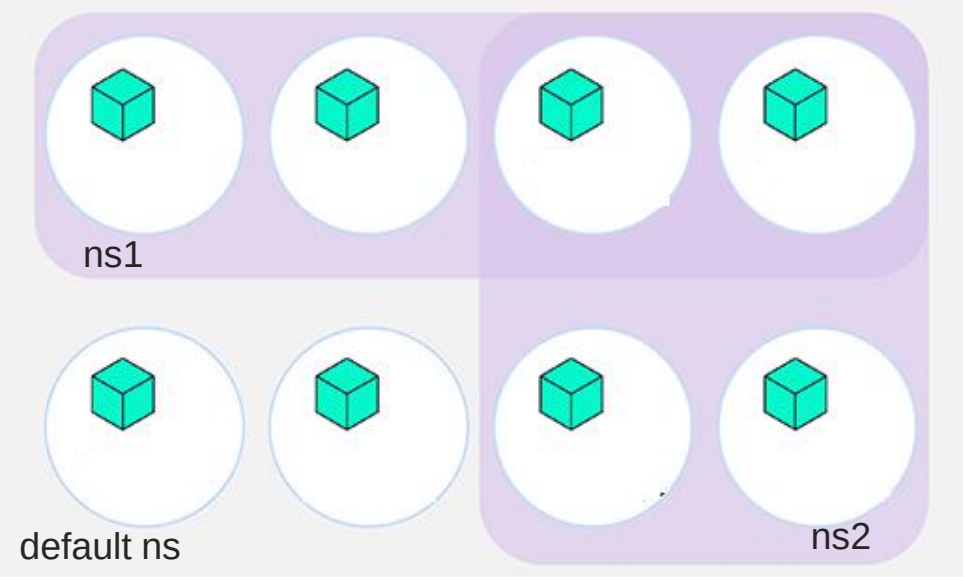
BACKUP SLIDES

Basic Kubernetes concepts / abstractions





Namespaces abstraction



Cluster view

Tests we run today

What tests we have in Gitlab CI?

- Clean environment – always from scratch
- Unit tests
  - Google tests unittests (link)
  - Old namespace and quarkDB
- Functional tests
  - System tests: ~450 functional tests (link) included fuse tests (link)
  - Runs as 'root' on MGM
  - Repeated for:
    - quarkDB
    - Old namespace
    - quarkDB with xroot (manual)
  - Tests are repeated in the Kubernetes infrastructure
- Stress tests
  - 1 machine - 10000 files – 1, 2, 10, 100 threads write/stat/read/delete
  - Tests are repeated in Kubernetes infrastructure

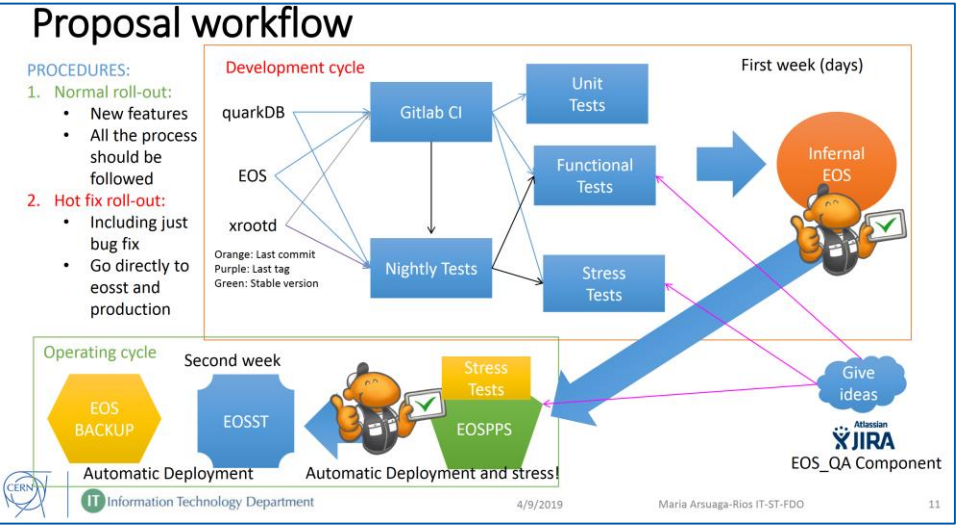
CERN | IT Information Technology Department 4/9/2019 Maria Arsuaga-Rios IT-ST-FDO

*

i.e.

```
$: kubectl exec -n myNamespace -it myMGM -- eos-instance-test-ci

$: Running MQ tests
… … …
```



Tests you can run (we can run)

i.e.

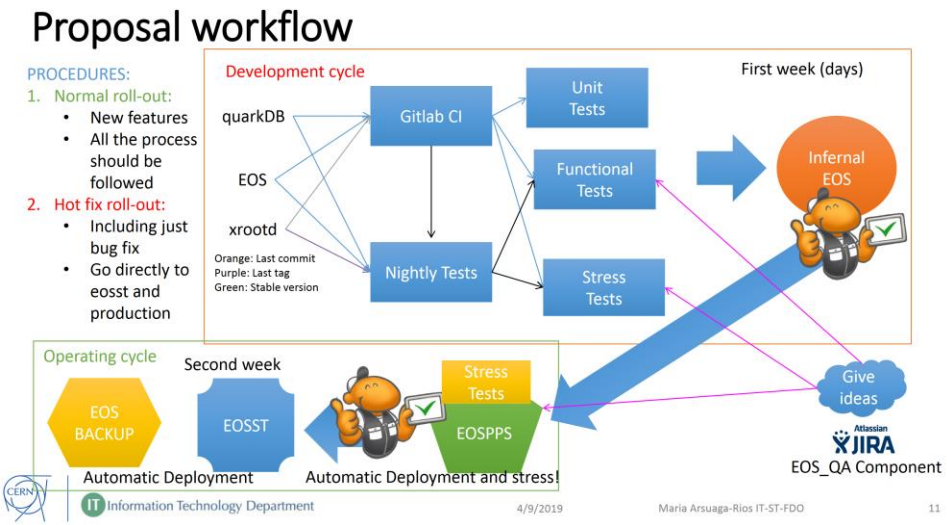
```
$: kubectl exec -n my_namespace -it $(get_podname eos-mgm) -- <test_you_like_most1>

$: kubectl exec -n my_namespace -it $(get_podname eos-cli1) -- <test_you_like_most2>

$: kubectl delete pod -n my_namespace $(get_podname eos-mgm)

$: kubectl delete pod -n my_namespace $(get_podname eos-fst8) # ( and see what happens )

$: kubectl exec -n my_namespace -it $(get_podname eos-qdb) -- <test_you_like_most3>
… … …
```



Tests you can run (we can run)

i.e.

```
# set a namespace.
$: kubectl config set-context $(kubectl config current-context) --namespace=<insert-namespace-name-here>

$: kubectl exec -it $(get_podname eos-mgm) -- <test_you_like_most1>
$: kubectl delete pod $(get_podname eos-mgm)
$: kubectl delete pod $(get_podname eos-fst8) # ( and see what happens )
$: kubectl exec -it $(get_podname eos-qdb) -- <test_you_like_most3>
… … …
```