Test execution for kui 8.12.2

Pre-requisites for kui test execution

The main pre-requisite for kui test execution is the availability of k8s cluster. In my case, I set up a minikube cluster. The availability of docker installation is an implicit requirement. Just in case, it's not available, steps to install docker 19.03.8 are documented here. Here are the steps that I used to setup minikube on RHEL 8.2 as root user:

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-
linux-ppc64le \
  && chmod +x minikube
mv minikube /usr/local/bin/
minikube start --driver=none
```

Test execution overview

In order to execute the tests, post the build-script success, the following commands need to be executed from the parent directory of kui source:

```
CWD=`pwd`
VERSION=v12.16.1
DISTRO=linux-ppc64le
PATH=$CWD/node-$VERSION-$DISTRO/bin:$PATH
cd kui/
npm run testv2
```

Also, one thing to note is that the test execution does not commence if **display :0** is not being used either by **X** or **vncserver**. It complains about being unable to detect chrome or about chrome experiencing crash. So, if you do not have **X** running on **display :0**, you need to start vncserver as:

```
yum install -y fluxbox tigervnc-server xterm
mkdir -p ~/.vnc
echo "fluxbox &"> ~/.vnc/xstartup
chmod u+x ~/.vnc/xstartup
vncserver :0 -rfbport 5901
```

In order to execute the tests belonging to a specific file, it's needed to set an environment variable named ***LAYER*** to the path of the file inside ***node_modules/@kui-shell/<plugin_name>/dist/test*** directory. The easy way to identify the value to be set for *LAYER,* for the tests marked in the table above, is to just use the path post ***test*** and replace ***.ts*** extension with ***.js***. e.g. To execute the tests from ***#74 (plugins/plugin-kubectl/src/test/k8s2/kustomize.ts)***, here are the commands:

```
export LAYER=k8s2/kustomize.ts
npm run testv2
```

Also, there are a few exceptions like ***plugins/plugin-kubectl/helm/src/test/helm/helm-repo-add-and-search.ts***, where ***src*** directory does not lie directly inside ***<plugin_xxx>*** directory. For being able to detect the tests in such cases, the following patch is needed:

```
diff --git a/packages/test/bin/runTest.sh b/packages/test/bin/runTest.sh
index bef9bf4f5..3d4b5d94b 100755
--- a/packages/test/bin/runTest.sh
+++ b/packages/test/bin/runTest.sh
@@ -61,7 +61,7 @@ if [ -n "$LAYER" ]; then
       if [[ $LAYER == *"core"* ]]; then
         TEST_SUITES=$(find -H "$TEST_SUITE_ROOT"/{plugin-*,core,client} -path
"*/dist/test/$LAYER" -o -path '*/core/test' -maxdepth 4)
       else
```

```
-         TEST_SUITES=$(find -H "$TEST_SUITE_ROOT"/{plugin-*,core,client} -path
"*/dist/test/$LAYER" -maxdepth 4)
+         TEST_SUITES=$(find -H "$TEST_SUITE_ROOT"/{plugin-*,core,client}/* -path
"*/dist/test/$LAYER" -maxdepth 4)
        fi
      else
        if [[ $LAYER == *"core"* ]]; then
```

Test failure analysis

As per the results of test execution on **RHEL 8.2** ppc64le, **764** tests passed, while **74** failed. The failing tests were skipped to complete the test execution. The table below lists the failing tests:

Sr No	Test file	Failure count	Failing Test's title
1	plugins/plugin-bash-like/src/test/bash-like/bash-like.ts	2	should rm that file
2			should rmdir a dir with spaces
3	plugins/plugin-client-common/src/test/editor/edit.ts	8	should rm ${nonExistFilePath2}
4			should copy the edit input
5			should edit but not save the content of an existing file
6			should re-open the file and see the unchanged content
7			should edit and save the content
8			should re-open the edited file and see the updated content
9			should edit, then paste, and still have focus
10			should have that pasted text after refresh
11	plugins/plugin-core-support/src/test/core-support2/history.ts	2	should hit the up arrow and see previous command
12			should hit the down arrow and see previous command
13	plugins/plugin-core-support/src/test/core-support2/tab-completion.ts	18	should tab complete the ../../packages/core directory
14			should tab complete file with spaces unique ${tmp1.name}
15			should tab complete file with spaces unique with dash option
16			should tab complete file with spaces unique with backslash escape
17			should tab complete file with spaces non-unique
18			should tab complete file with spaces non-unique with dash option
19			should tab complete file with spaces non-unique with backslash escape
20			should tab complete file with spaces unique with backslash escape variant 2
21			should complete on the single-entry directory (tab completion of directories)
22			should complete on the single-entry directory (auto-completing the single entry in the directory)
23			should complete on a dot file
24			should tab complete local file path with options then click on second
25			should tab complete local file path with options, expect prompt update

26			should tab complete local file path with options then click on first
27			should tab complete the data directory
28			should tab complete the data/core/empty.js file
29			should tab complete local file path
30			should tab complete local file path, then options go away on edit
31	plugins/plugin-core-support/src/test/core-support2/tab-navigation.ts	4	should focus on repl input since we just hit Enter
32			should tab to the ${selector} hitEnter=${hitEnter}
33			should be the beginning of a full cycle
34			should be the end of the full cycle
35	plugins/plugin-kubectl/helm/src/test/helm/helm-repo-add-and-search.ts	4	should add a helm repo
36			should list helm repos
37			should search for ${desiredImage}
38			should remove a helm repo
39	plugins/plugin-kubectl/helm/src/test/helm/helm.ts	14	should show 500 error for helm help --tls
40			should show 500 error for helm get
41			should show 500 error for helm create
42			should show 500 error for helm install
43			should show 500 error for helm delete
44			should list empty releases via helm ${list}
45			should create sample helm chart
46			should refresh as a quick way to close the sidecar
47			should show history
48			should list that new release via helm list
49			should list that new release via helm list
50			should show the release in sidecar via helm get
51			should delete sample helm chart
52			should list empty releases via helm list again
53	plugins/plugin-kubectl/logs/src/test/logs/logs-dash-c-via-table.ts	2	should wait for the pod to come up
54			should follow the logs
55	plugins/plugin-kubectl/logs/src/test/logs/logs-dash-f-via-table.ts	2	should wait for the pod to come up
56			should follow the logs
57	plugins/plugin-kubectl/logs/src/test/logs/logs-via-table.ts	3	should wait for the pod ${podName} to come up
58			should show logs for pod ${podName} container ${containerName}
59			should click retry button
60	plugins/plugin-kubectl/src/test/k8s-popup/a-ibmcloud-plugin.ts	1	should get default namespace via ibmcloud kui ${kubectl}
61	plugins/plugin-kubectl/src/test/k8s-popup/headless-create-pod.ts	1	should delete the namespace ${ns}
62	plugins/plugin-kubectl/src/test/k8s1/apply-crd.ts	1	should delete the custom resource definition from URL via ${command}
63	plugins/plugin-kubectl/src/test/k8s1/deployment.ts	5	should create deployment from local file
64			should list deployments
65			should list pods in deployment, then navigate using Show Owner Reference button

66			should delete the deployment by name
67			should delete the deployment by clicking on the sidecar delete button
68	plugins/plugin-kubectl/src/test/k8s1/edit.ts	1	should modify the content, introducing a ${title}
69	plugins/plugin-kubectl/src/test/k8s2/job.ts	2	should create a job
70			should delete a job
71	plugins/plugin-kubectl/src/test/k8s2/kubectl-exec-vi.ts	3	should wait for the pod to come up
72			should use kubectl exec vi through pty
73			should use kubectl exec to cat the file we just edited
74	plugins/plugin-kubectl/src/test/k8s2/kustomize.ts	1	should create deployment from local kustomize directory via ${command} ${verb} ${dashK} expecting ${expecting}

Initial analysis of these test failures included independent execution of the tests belonging to each file and comparison of the results with those on RHEL 7.7 intel (RH7 since RH8 intel VM wasn't available. but the results didn't differ much between RH7 power and RH8 power, so we're good there). With this comparison, I was able to establish parity with intel for 43 tests. Here are the notes for those:

1. #1 to #9, #44, #45, #47 to #52, #68 failed on both.
2. #10 was not reproducible anymore on power or intel.
3. #11 to #34, pass on intel as well as power on independent execution. Fail on both on full test suite execution.

So, 31 test failures needed further analysis and debugging. Here is the categorization and details of the further analysis of these 31 tests:

1. **helm**: #35 to #43, #46 were related to missing **helm** setup (including tiller pod) on power.
   - Completed helm setup along with tiller pod on power with the following steps:

```
wget https://get.helm.sh/helm-v2.16.1-linux-ppc64le.tar.gz
tar xzf helm-v2.16.1-linux-ppc64le.tar.gz
cd linux-ppc64le/
cat > Dockerfile << EOF
FROM registry.access.redhat.com/ubi8

RUN yum install -y ca-certificates socat

ENV HOME /tmp

COPY helm /helm
COPY tiller /tiller

EXPOSE 44134
USER 65534
ENTRYPOINT ["/tiller"]
EOF
docker build -t helm/tiller-ppc64le:v2.16.1 .
cp helm /usr/local/bin/
kubectl create serviceaccount --namespace kube-system tiller
kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-
admin --serviceaccount=kube-system:tiller
kubectl --namespace kube-system patch deploy tiller-deploy -p
'{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}'
```

   - With this setup, all helm tests including #44, #45, #47 to #52 passed on power.

2. **jare/alpine-vim**: #53 to #59, #71 to #73 were due to missing **jare/alpine-vim:latest** image for power.

- Created the image locally by modifying the dockefile @ https://hub.docker.com/r/jare/alpine-vim/dockerfile to:

```
# Multistage builds to reduce image size to ~37MB
# by tuanhtrng
FROM ppc64le/alpine:latest as builder

MAINTAINER JAremko <w3techplaygound@gmail.com>

WORKDIR /tmp

# Install dependencies
RUN apk add --no-cache \
    build-base \
    ctags \
    git \
    libx11-dev \
    libxpm-dev \
    libxt-dev \
    make \
    ncurses-dev \
    python2 \
    python2-dev

# Build vim from git source
RUN git clone https://github.com/vim/vim \
&& cd vim \
&& ./configure \
    --disable-gui \
    --disable-netbeans \
    --enable-multibyte \
    --enable-pythoninterp \
    --with-features=big \
    --with-python-config-dir=/usr/lib/python2.7/config \
&& make install

FROM ppc64le/alpine:latest

COPY --from=builder /usr/local/bin/ /usr/local/bin
COPY --from=builder /usr/local/share/vim/ /usr/local/share/vim/
# NOTE: man page is ignored

RUN apk add --no-cache \
    diffutils \
    libice \
    libsm \
    libx11 \
    libxt \
    ncurses

ENTRYPOINT ["vim"]
```

- But the tests were still failing on power, because the image was explicitly being pulled from docker.io. So, had to make a minor change in the deployment file:

```
diff --git a/plugins/plugin-kubectl/tests/data/k8s/kubectl-exec.yaml
b/plugins/plugin-kubectl/tests/data/k8s/kubectl-exec.yaml
index 841471c84..1a14507a1 100644
--- a/plugins/plugin-kubectl/tests/data/k8s/kubectl-exec.yaml
+++ b/plugins/plugin-kubectl/tests/data/k8s/kubectl-exec.yaml
@@ -5,6 +5,7 @@ metadata:
spec:
   containers:
   - image: jare/alpine-vim
+    imagePullPolicy: IfNotPresent
     command:
       - /bin/sh
       - "-c"
```

3. **full-suite vs independent:** #61, #62, #69, #70 had failed on power on full test suite execution. But passed on both on independent execution.
    - #61 turned out to be a flaky test for which the failure was not seen on power even on full test suite execution this time.
    - #62, #69, #70, however, were reproducible on power on full test suite execution. Post that even independent execution exhibited the issue until minikube was restarted. So, this seemed more related to the unstable cluster rather than kui test issue.
    - So, ignored these.

4. **rvennam/drone-app:** #63 to #67 failed due to missing **rvennam/drone-app:latest** image for power.
    - Tried to find dockerfile or other alternative image on dockerhub to no avail. Also, failed to find source on github.
    - Checked the nature of the tests. They seemed to only need an application image that starts and does something (sort of infinite loop) until stopped and they're just related to creation/deletion/listing of pods.
    - Replaced "rvennam/drone-app:latest" with "jare/alpine-vim:latest" and got the tests to PASS. Here is the change needed to replace the image:

```
diff --git a/plugins/plugin-kubectl/tests/data/k8s/deployment.yaml
b/plugins/plugin-kubectl/tests/data/k8s/deployment.yaml
index 5f05bdac2..2e538814a 100644
--- a/plugins/plugin-kubectl/tests/data/k8s/deployment.yaml
+++ b/plugins/plugin-kubectl/tests/data/k8s/deployment.yaml
@@ -27,7 +27,7 @@ spec:
                 topologyKey: "kubernetes.io/hostname"
       containers:
       - name: drone-app
-        image: rvennam/drone-app:latest
+        image: monopole/hello:1
         # imagePullPolicy: Always
         ports:
         - containerPort: 3000
```

5. **monopole/hello:** #74 failed due to missing **monopole/hello:1** image for power.
    - As per the instructions from https://github.com/monopole/hello/blob/master/containerize.md, created a script build.bash. Contents below:

```
#!/bin/bash

function buildVersionedExecutable {
  local tmpDir=$1
  local githubUser=$2
  local pgmName=$3
  local version=$4

  local package=github.com/${githubUser}/${pgmName}
  local newPgm=$tmpDir/${pgmName}_${version}

  GOPATH=$tmpDir go get -d $package

  cat $tmpDir/src/${package}/${pgmName}.go |\
      sed 's/version = 0/version = '${version}'/' \
      >${newPgm}.go

  echo Compiling ${newPgm}.go

  GOPATH=$tmpDir CGO_ENABLED=0 GOOS=linux go build \
      -o $tmpDir/${pgmName} \
      -a -installsuffix cgo ${newPgm}.go
}
```

```
function runAndQuitRawBinaryToTest {
  local tmpDir=$1
  local pgmName=$2
  local port=$3

  echo Running server $tmpDir/$pgmName
  ALT_GREETING=salutations \
      $tmpDir/$pgmName --enableRiskyFeature --port $port &

  # Let it get ready
  sleep 2

  # Dump html to stdout
  curl --fail --silent -m 1 localhost:$port/godzilla

  # Send query of death
  curl --fail --silent -m 1 localhost:$port/quit
  echo Server stopped
}

function buildDockerImage {
  local tmpDir=$1
  local pgmName=$2
  local version=$3

  # Repo holds just one image, give repo same name as image.
  local dockerRepo=$pgmName

  local dockerFile=$tmpDir/Dockerfile
  cat <<EOF >$dockerFile
FROM scratch
ADD $pgmName /
CMD ["/$pgmName"]
EOF
  echo Docker build
  docker build -t $dockerRepo:$version -f $dockerFile $tmpDir
  echo End docker build
}

function runAndQuitInsideDockerToTest {
  local pgmName=$1
  local version=$2
  local port=$3

  echo Docker run, mapping $port to internal 8080
  docker run -d -p $port:8080 $pgmName:$version
  sleep 3
  docker ps | grep $pgmName

  echo Requesting docker server
  curl -m 1 localhost:$port/kingGhidorah
  curl -m 1 localhost:$port/quit
}

function pushToDockerHub {
  local dockerUser=$1
  local pgmName=$2
  local version=$3
  local repoName=$pgmName
  local id=$(docker images |\
      grep $pgmName | grep " $version " | awk '{printf $3}')

  docker tag $id $dockerUser/$repoName:$version
  docker push $dockerUser/$repoName:$version
}
```

```
function buildContainer {
  local githubOrg=$1
  local pgmName=$2
  local version=$3
  local testPort=$4
  local tmpDir=$(mktemp -d)

  echo tmpDir=$tmpDir
  buildVersionedExecutable $tmpDir $githubOrg $pgmName $version
  runAndQuitRawBinaryToTest $tmpDir $pgmName $testPort

  buildDockerImage $tmpDir $pgmName $version
  docker images --no-trunc | grep $pgmName
  sleep 4
  runAndQuitInsideDockerToTest $pgmName $version $testPort
}

function removeLocalImage {
  local pgmName=$1
  local version=$2

  echo docker rmi $pgmName:$version
  docker rmi $pgmName:$version
  id=$(docker images | grep $pgmName | grep " $version " | awk '{printf $3}')
  echo docker rmi -f $id
  docker rmi -f $id
}

buildContainer monopole hello 1 8999
```

- And executed the following steps to create the image locally, post which the tests passed:

```
chmod +x build.bash
./build.bash
docker tag hello:1 monopole/hello:1
```

6. **Ibmcloud**: #60 needed **ibmcloud** setup for further analysis.
   - Used the steps from the following links to install ibmcloud cli and to install kui as ibmcloud plugin:
     https://cloud.ibm.com/docs/cli?topic=cli-install-ibmcloud-cli#install-ibmcloud-cli
     https://github.com/IBM/kui/tree/master/tools/go/ibmcloud
   - Steps used:

```
wget https://golang.org/dl/go1.15.3.linux-ppc64le.tar.gz
tar xzf go1.15.3.linux-ppc64le.tar.gz
export PATH=$PATH:`pwd`/go/bin
wget https://clis.cloud.ibm.com/download/bluemix-cli/1.2.3/linux64/archive
tar xzf archive
export PATH=$PATH:`pwd`/IBM_Cloud_CLI/
cd kui/tools/go/ibmcloud/
make
make install
[root@p00XvmYY kui]# ibmcloud plugin show kui

Plugin Name                                kui
Plugin Version                             8.12.2
Plugin SDK Version                         0.4.0
Minimal IBM Cloud CLI version required     N/A

Commands:
kui    Kui Visual Terminal
```

   - But the following commands failed on power and produced no result on intel:

```
ibmcloud kui kubectl get pods -A
ibmcloud kui kubectl get default ns
```

   - Raised an issue with the community @ https://github.com/IBM/kui/issues/6095 , but received no response on that. Since the behaviour is in parity on both platforms, not pursuing this anymore.