**Visual Flow installation guide**

Contents

[About 3](#_Toc128743150)

[Pre-requisites: 4](#_Toc128743151)

[1. Kubernetes metrics server 4](#_Toc128743152)

[2. Argo workflow 5](#_Toc128743153)

[3. Redis database 9](#_Toc128743154)

[4. Setup authentication methods 10](#_Toc128743155)

[4.1 Set up authentication via GitHub 10](#_Toc128743156)

[4.2 Set up OICD authentication via Keycloak (optional) 10](#_Toc128743157)

[Installation 14](#_Toc128743158)

[1. Deploying secrets 14](#_Toc128743159)

[2. Deploying vf-app chart 15](#_Toc128743160)

[Links & Additional information 22](#_Toc128743161)

# About

Visual Flow is an ETL tool designed for effective data manipulation via convenient and user-friendly interface. The tool has the following capabilities:

* Can integrate data from heterogeneous sources: DB2, IBM COS, AWS S3, Elastic Search, Clickhouse, PostgreSQL, MySQL/Maria, MSSQL, Oracle, Cassandra, Mongo, Redis, Redshift
* Leverage direct connectivity to enterprise applications as sources and targets
* Perform data processing and transformation
* Run custom code
* Leverage metadata for analysis and maintenance

This guide is created to help you in deployment Visual Flow application on the local kubernetes environment. As test environment we use Minikube installed on the laptop.

# Pre-requisites:

The application requires the following services installed on kubernetes cluster:

* [Kubernetes metrics server](https://kubernetes.io/docs/tasks/debug-application-cluster/resource-metrics-pipeline/)
* [Argo Workflows](https://github.com/argoproj/argo-workflows)
* Redis
* Keycloak (optional)

## **Kubernetes metrics server**

Some clusters include default server metrics deployment. To check whether the metrics server is running on your cluster, run the following command:

|  |
| --- |
| kubectl get pods --all-namespaces | grep metrics-server |

If the metrics server is running, then in the response, you will see information about the running nodes:

|  |
| --- |
| NAME CPU(cores) MEMORY(bytes)  coredns-565d847f94-hbwjz 2m 51Mi  etcd-minikube 16m 50Mi  kube-apiserver-minikube 28m 375Mi |

Otherwise, run the following command to install the latest version of server metrics.

|  |
| --- |
| kubectl apply -f \  https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml |

After setting server metrics, you can get metrics for any node or pod using the *kubectl get* tool. Use the following commands to get metrics for all nodes and pods.

|  |
| --- |
| # Get the metrics for all nodes  kubectl get --raw /apis/metrics.k8s.io/v1beta1/nodes  # Get the metrics for all pods  kubectl get --raw /apis/metrics.k8s.io/v1beta1/pods |

You can disable ssl using --kubelet-insecure-tls key. Insert it to the deployment section of the components.yaml manifest file.

|  |
| --- |
| …  spec:  containers:  - args:  - --cert-dir=/tmp  - --secure-port=4443  - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname  - --kubelet-use-node-status-port  - --metric-resolution=15s  - --kubelet-insecure-tls  … |

For additional information you can visit: [https://kubernetes.io/docs/tasks/debug/debug-cluster/resource-metrics-pipeline/, https://kubernetes-sigs.github.io/metrics-server/](https://kubernetes.io/docs/tasks/debug/debug-cluster/resource-metrics-pipeline/,%20https:/kubernetes-sigs.github.io/metrics-server/)

## **Argo workflow**

Argo Workflows is an open source container-native workflow engine for orchestrating parallel jobs on Kubernetes. To install Argo Workflows, navigate to the [releases page](https://github.com/argoproj/argo-workflows/releases), find the release you wish to use (the latest full release is preferred) and install according instructions provided.

We installed Argo workflow application using helm. Actual helm chart can be downloaded from [charts-argo-workflows](https://github.com/argoproj/argo-helm/tree/main/charts/argo-workflows).

Installation steps:

|  |
| --- |
| kubectl create namespace argo-workflow |

* Create namespace for argo-workflow application
* Download helm chart

|  |
| --- |
| helm repo add argo <https://argoproj.github.io/argo-helm>  helm pull argo/argo  cd argo |

* Install application

|  |
| --- |
| helm install argo . -f values.yaml -n argo-workflow |

* Create roles

|  |
| --- |
| kubectl apply -f - <<EOF  apiVersion: rbac.authorization.k8s.io/v1  kind: Role  metadata:  name: argo-role  namespace: argo-workflow  rules:  - apiGroups:  - ""  resources:  - configmaps  verbs:  - get  - watch  - list  - create  - patch  - apiGroups:  - ""  resources:  - secrets  verbs:  - get  - apiGroups:  - ""  resources:  - pods  - pods/exec  - pods/log  verbs:  - get  - list  - watch  - create  - patch  - apiGroups:  - ""  resources:  - events  verbs:  - watch  - create  - apiGroups:  - ""  resources:  - serviceaccounts  verbs:  - get  - list  - watch  - create  - patch  - apiGroups:  - argoproj.io  resources:  - eventsources  - sensors  - workflows  - workfloweventbindings  - workflowtemplates  - cronworkflows  - cronworkflows/finalizers  - clusterworkflowtemplates  verbs:  - get  - list  - watch  - create  - patch  EOF |

* Create service account

|  |
| --- |
| kubectl create sa argo-rw -n argo-workflow |

* Create role bindings

|  |
| --- |
| kubectl create rolebinding argo-rw --role=argo-rw \  --serviceaccount=argo-workflow:argo-rw -n argo-workflow |

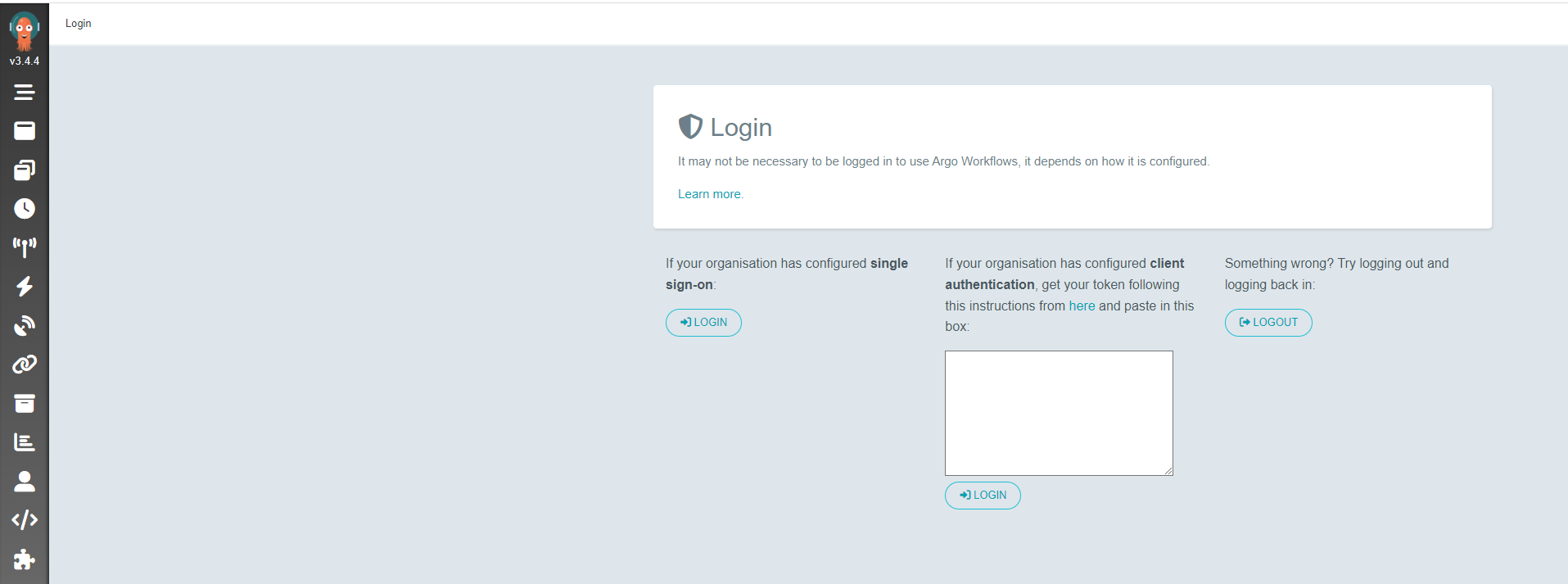
* Create secret token

|  |
| --- |
| kubectl apply -f - <<EOF  apiVersion: v1  kind: Secret  metadata:  name: argo-rw.service-account-token  annotations:  kubernetes.io/service-account.name: argo-rw  type: kubernetes.io/service-account-token  EOF |

To access Argo workflow UI you can use port forwarding:

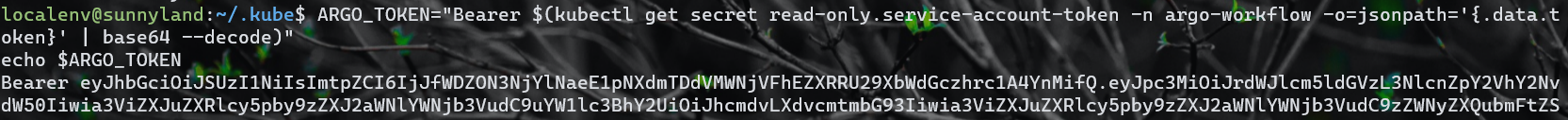
|  |
| --- |
| kubectl -n argo-workflow port-forward svc/<argo\_service\_name> 2746:2746 |

and use link <http://localhost:2746/login>. You will be redirected to the Argo login page

As access token use the command output:

|  |
| --- |
| echo "Bearer $(kubectl get secret argo-rw.service-account-token -n \  argo-workflow -o=jsonpath='{.data.token}' | base64 --decode)" |

It should be like:



## **Redis database**

Installation steps:

* Download helm charts

|  |
| --- |
| helm repo add bitnami <https://charts.bitnami.com/bitnami>  helm pull bitnami/redis |

* Create namespace

|  |
| --- |
| kubectl create namespace redis |

* Install Redis database

|  |
| --- |
| helm install redis . -f values.yaml -n redis |

* Check database availability

|  |
| --- |
| export REDIS\_PASSWORD=$(kubectl get secret --namespace redis redis \  -o jsonpath="{.data.redis-password}" | base64 -d)  export NODE\_IP=$(kubectl get nodes --namespace redis \  -o jsonpath="{.items[0].status.addresses[0].address}")  export NODE\_PORT=$(kubectl get --namespace redis \  -o jsonpath="{.spec.ports[0].nodePort}" services redis-master)  REDISCLI\_AUTH="$REDIS\_PASSWORD" redis-cli -h $NODE\_IP -p $NODE\_PORT |

You will see the next output in case Redis database is available:

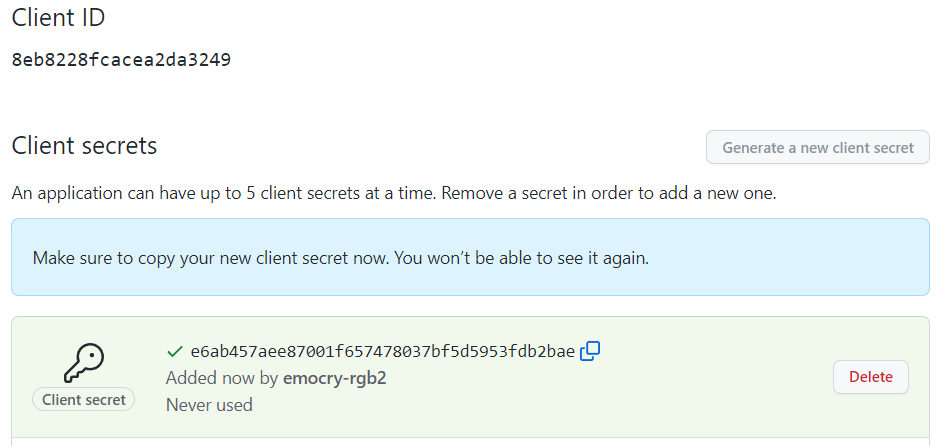
|  |
| --- |
| 172.18.109.149:30907> CONFIG GET databases  1) "databases"  2) "16"  172.18.109.149:30907> |

## **Setup authentication methods**

Visual Flow application support some authentications methods: OICD, GitHub authentication.

### **4.1 Set up authentication via GitHub**

* Log in to your GitHub account.
* Create new OAuth app. In your GitHub account : move Settings > Developer settings > OAut apps > New OAuth app.

****

### **Set up OICD authentication via Keycloak (optional)**

* + 1. **Keycloak installation:**
* Download helm charts

|  |
| --- |
| helm pull bitnami/keycloak |

* Create namespace

|  |
| --- |
| kubectl create namespace keycloak |

* Modify helm charts if necessary

To access keycloak UI modify helm chart: change service type to NodePort

|  |
| --- |
| …  service:  type: NodePort  http:  enabled: true  ports:  http: 80  https: 443  nodePorts:  http: 31765  https: 31768  … |

* Install keycloak using helm

|  |
| --- |
| helm install kk bitnami/keycloak -n keycloak |

* Check installation

Run the following commands

|  |
| --- |
| export HTTP\_NODE\_PORT=$(kubectl get --namespace keycloak -o jsonpath="{.spec.ports[?(@.name=='https')].nodePort}" services kk-keycloak)  export NODE\_IP=$(kubectl get nodes --namespace keycloak -o jsonpath="{.items[0].status.addresses[0].address}")  echo http://${NODE\_IP}:${HTTP\_NODE\_PORT}/ |

Output will be like: http://172.18.109.149:31141/

Insert this address to your browser and you will be taken to the control panel.

To get password execute command:

|  |
| --- |
| kubectl get secret <keycloak-secret-name> -o jsonpath='{.data}' -n keycloak |

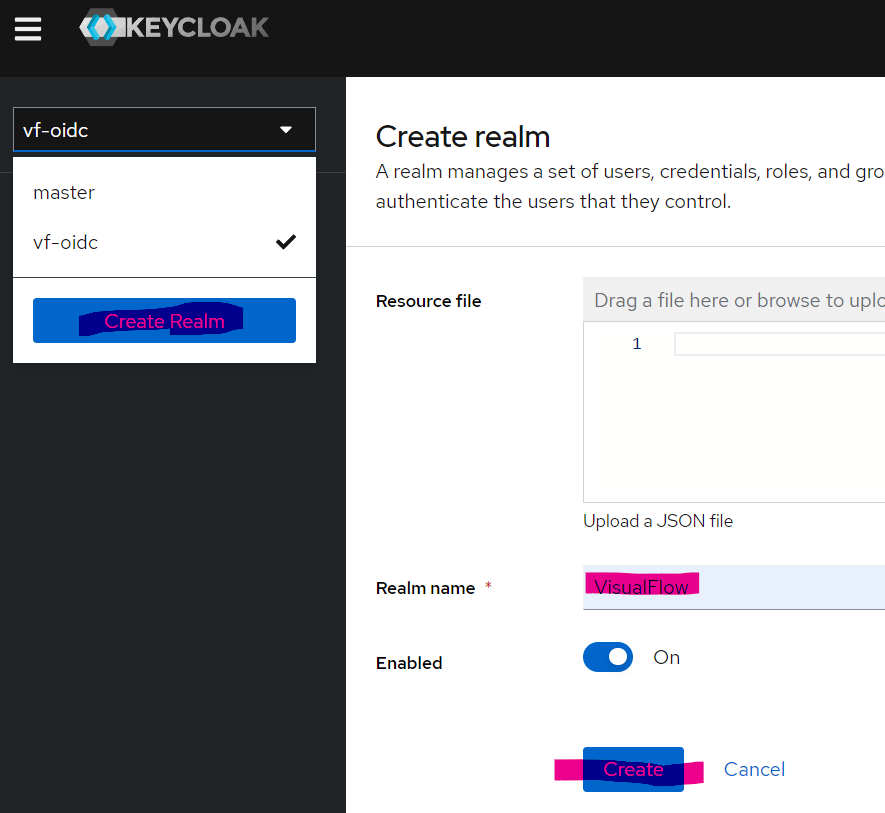
The output is similar to:

|  |
| --- |
| {"admin-password":"UkR5R2hKVEdkSg=="} |

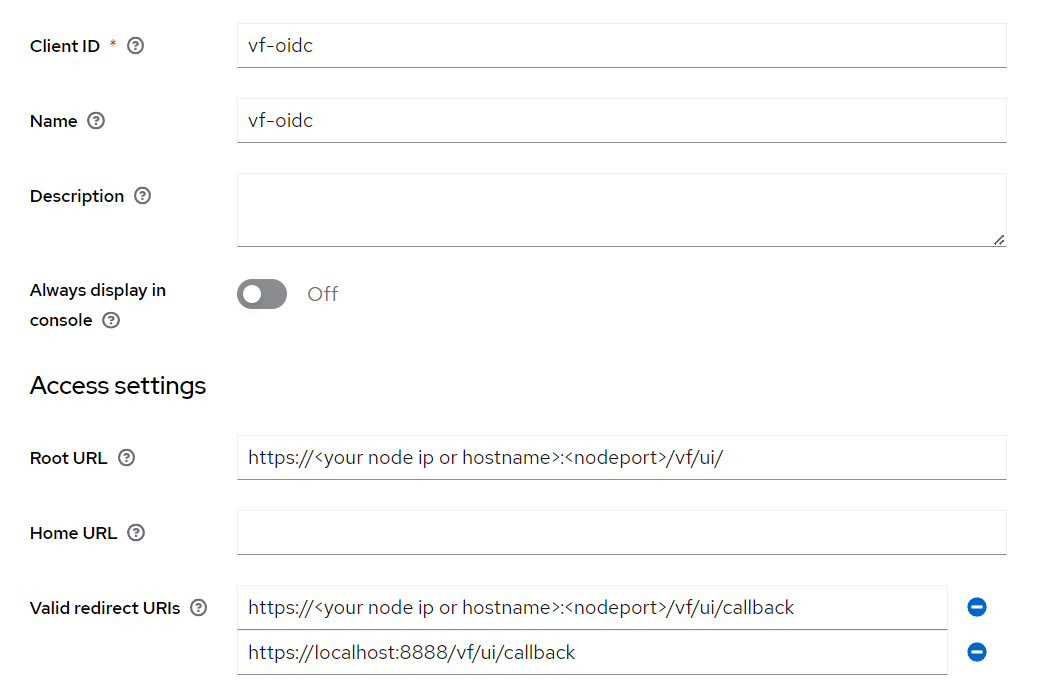
Decode the password data:

|  |
| --- |
| echo 'UkR5R2hKVEdkSg==' | base64 –decode |

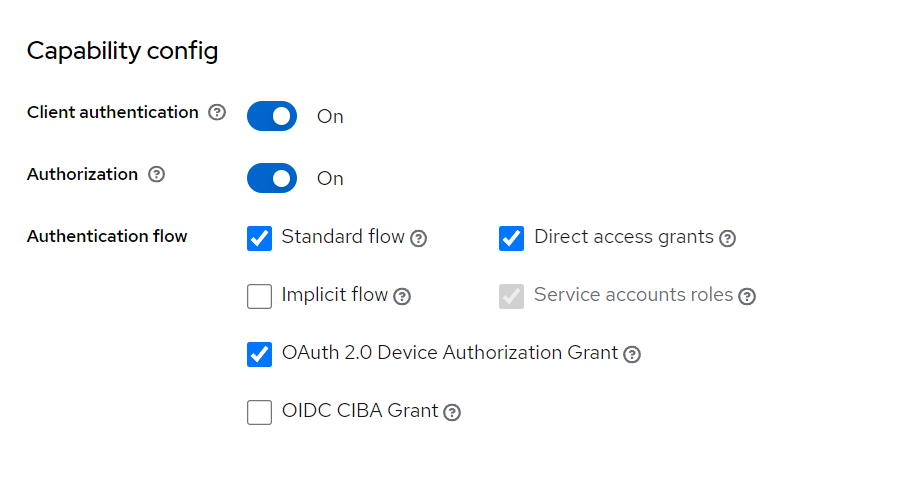
* + 1. **Keycloak configuration**
* Login into keycloak admin console and press CREATE REALM, enter realm name for example VisualFlow and then CREATE button

****

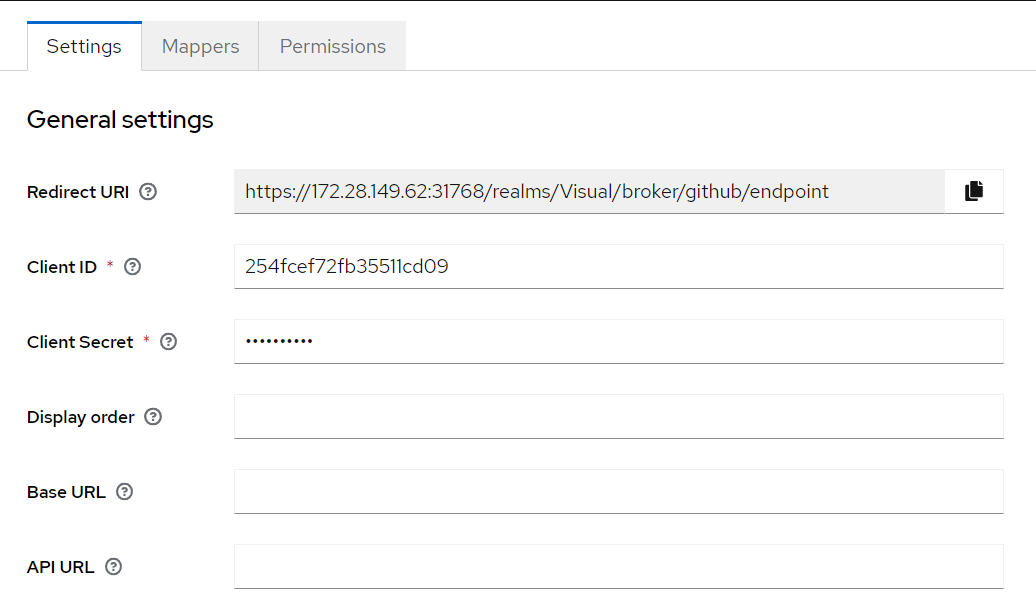
* Press CLIENTS > CREATE CLIENT > vf-oidc. We use NodeIP and NodePort but you can put just app hostname

****

* Tune Capability config set the following checkboxes

****

* Configure realm settings. Just leave it without changes. At the bottom of the page press OpenID Endpoint Configuration. You will be redirected to an array with parameters, some of which are used to set up Visual Flow application's helm charts

****

* Setup your GitHub OAuth application. Go to keycloak admin console > Identity providers > Add provider > GitHub

# Installation

The helm chart is the recommended way to install and configure Visual flow on Kubernetes. To get the latest version of the charts, clone the repository [VF-deploy](https://github.com/ibagroup-eu/VF-deploy.git) to your local machine.

### **Deploying secrets**

Visual Flow application requires TLS certificates and Docker repository that have to be deployed as kubernetes secrets. Use vf-secret chart to deploy them.

Installation steps:

* Generate keys

For key generation the next command can be used:

|  |
| --- |
| openssl genrsa -out ca.key 2048  openssl req -x509 \  -new -nodes \  -days 365 \  -key ca.key \  -out ca.crt \  -subj "/CN=vf-app-backend" |

Run it twice to generate certificates for backend and frontend services. Upon completion, you should have 4 files. Store key pairs in different folders.

* Prepare values.yaml

|  |
| --- |
| vf-secret/values.yaml file  project: visual-flow  secrets:  - name: vf-backend-ssl  app: vf-app  type: kubernetes.io/tls  stringData:  tls.crt: |-  -----BEGIN CERTIFICATE-----  **(INSERT backend ca.crt file content)**  -----END CERTIFICATE-----  tls.key: |-  -----BEGIN PRIVATE KEY-----  (INSERT backend ca.key file content)  -----END PRIVATE KEY-----  - name: vf-frontend-ssl  app: vf-app  type: kubernetes.io/tls  stringData:  tls.crt: |-  -----BEGIN CERTIFICATE-----  **(INSERT frontend ca.crt content)**  -----END CERTIFICATE-----  tls.key: |-  -----BEGIN RSA PRIVATE KEY-----  (INSERT frontend ca.crt content)  -----END RSA PRIVATE KEY-----  - name: vf-image-pull  app: vf-app  type: kubernetes.io/dockerconfigjson  stringData:  .dockerconfigjson: |-  {  "auths": {  "**<your docker registry>**": {  "auth": "**<BASE64 ENCODED USER:PASS>**"  }  }  } |

### **Deploying vf-app chart**

* Prepare values.yaml

Example of the values file:

|  |
| --- |
| # see description of parameters in ./README.md file.  project: visual-flow  imagePullSecret: vf-image-pull **# put your own image pull secret**  # backend part of app  backend:  createRoles: true  serviceAccount:  name: vf-app-backend-user  addSA: true  deployment:  replicas: 1  image:  repository: ibagomel/visual-flow-backend **# put path to your repo**  tag: latest  pullPolicy: IfNotPresent  # cmd: "sh ./generate\_keystore\_p12.sh; java -Xms1g -Xmx8g -jar vf-api.jar --spring.config.location=file:/config/application.yaml"  # variables:  secretVariables:  KEYSTORE\_PASS: "<PASS>"  SLACK\_API\_TOKEN: "<TOKEN>"  sslSecret: vf-backend-ssl  # resources: {}  configFile:  oauth:  userInfo: "https://api.github.com/user" # "https://<GITLAB\_URL>/api/v4/user"  fieldsMap:  id: id  username: login # username  name: name  email: email  superusers:  - "ASkrytski@ibagroup.eu"  - "emocry-rgb2"  - "skrytski-ibagroup-eu"  notifications:  slack:  botToken: BOT\_TOKEN  channelName: CHANNEL\_NAME  mail:  default-encoding: UTF-8 # Default-encoding  host: smtp.gmail.com # An IP address or FQDN of the SMTP server  username: EMAIL # The user name that will be used to connect to the SMTP server.  password: APP\_PASSWORD # The password of the SMTP user  port: 587 # Usually 465 or 587.  properties:  mail:  debug: false  smtp:  debug: false  auth: true  starttls: true  protocol: smtp  test-connection: false  namespace:  label: vf-app  prefix: "${namespace.label}-"  # annotations:  # "[openshift.io/sa.scc.mcs]": "s0:c25,c20"  # "[openshift.io/sa.scc.supplemental-groups]": "1000/10000"  # "[openshift.io/sa.scc.uid-range]": "1000/10000"  pvc:  memory: 1Gi  mountPath: "/files"  argo:  serverUrl: "http://10.103.26.206:2746"  limits:  cpu: 0.5  memory: 512Mi  requests:  cpu: 0.1  memory: 64Mi  ttlStrategy:  secondsAfterCompletion: 5  secondsAfterSuccess: 5  secondsAfterFailure: 360  redis:  host: redis-master.redis  port: 6379  # username: ${REDIS\_USER}  database: 0  sparkJob:  repository: ibagomel/visual-flow-spark-job  tag: latest  jobSA: vf-spark-job  jobRB: vf-spark-job-edit  addSA: true  # kubernetesAPI: "k8s://https://kubernetes.default.svc"  slackJob:  repository: ibagomel/visual-flow-slack-notification  tag: latest  appAPItoken: "${SLACK\_API\_TOKEN}"  service:  type: NodePort  port: 8080  nodePort: 30918  annotations:  service.beta.openshift.io/serving-cert-secret-name: vf-backend-ssl  subPath: "/vf/be"  external:  enabled: true  type: ingress # route  host: vf-app-backend  annotations:  nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"  ssl:  termination: reencrypt  additionalFields:  insecureEdgeTerminationPolicy: Redirect  frontend:  deployment:  replicas: 1  image:  repository: ibagomel/visual-flow-frontend  tag: latest  pullPolicy: Always  variables:  # API\_SERVER: "https://172.28.149.62:30918/vf/be/"  # AUTHORIZATION\_URL: "https://172.28.149.62:31768/realms/Visual/protocol/openid-connect/auth"  # CALLBACK\_URL: "https://172.28.149.62:30917/vf/ui/callback"  # ISSUER\_URL: "https://172.28.149.62:31768/realms/Visual"  # LOGOUT\_URL: "https://172.28.149.62:31768/realms/Visual/protocol/openid-connect/logout"  # OIDC\_AVATAR\_KEY: avatar\_url  # OIDC\_AVATAR\_URL: 'https://api.github.com/users/${USERNAME}'  # STRATEGY: OIDC  # TOKEN\_URL: "https://172.28.149.62:31768/realms/Visual/protocol/openid-connect/token"  # USERINFO\_URL: "https://172.28.149.62:31768/realms/Visual/protocol/openid-connect/userinfo"  ##Github auth method  API\_SERVER: "https://172.21.192.14:30918/vf/be/" #https://vf-dev.apps.okd4.okd.gomel.iba.by/vf/be/  CALLBACK\_URL: "https://172.21.192.14:30917/vf/ui/callback"  STRATEGY\_CALLBACK\_URL: "https://172.21.192.14:30917/vf/ui/callback"  STRATEGY: GITHUB  GITHUB\_STRATEGY\_BASE\_URL: "https://github.com"  ISSUER\_URL: "https://github.com"  AUTHORIZATION\_URL: "https://github.com/login/oauth/authorize"  TOKEN\_URL: "https://github.com/login/oauth/access\_token"  LOGOUT\_URL: "https://172.21.192.14:30917/vf/ui/logout?redirect\_uri=https://172.21.192.14:30917/vf/ui/callback"  USERINFO\_URL: "https://172.21.192.14:30917/vf/ui/realms/vf-oidc/protocol/openid-connect/userinfo"  OIDC\_AVATAR\_KEY: avatar\_url  OIDC\_AVATAR\_URL: "https://api.github.com/users/${USERNAME}"  secretVariables:  CLIENT\_ID: |-  vf-oidc  CLIENT\_SECRET: |-  wDIJ0tHvnaAkFRSYT0QQfXiShh9dolst  SESSION\_SECRET: |-  a597d68350b02f2d  REDIS\_PASSWORD: |-  x6Bvxxh4hI  GITHUB\_APP\_ID: |-  254fcef72fb35511cd09  GITHUB\_APP\_SECRET: |-  e073efe08970eccd041187fd7374b0755d9d0eb7  REDIS\_HOST: |-  redis-master.redis  REDIS\_PORT: |-  6379  REDIS\_DB: |-  0  sslSecret: vf-frontend-ssl  resources: {}  service:  type: NodePort  port: 8888  nodePort: 30917  annotations:  service.beta.openshift.io/serving-cert-secret-name: vf-frontend-ssl  subPath: "/vf/ui/"  external:  enabled: true  type: ingress # route  host: vf-app-frontend  annotations:  nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"  ssl:  termination: Reencrypt  additionalFields:  insecureEdgeTerminationPolicy: Redirect  # dbservice:  # deployment:  # replicas: 1  # image:  # repository: ibagomel/vf-backend-db-service  # tag: 1  # pullPolicy: Always #IfNotPresent  # # cmd: "sh ./generate\_keystore\_p12.sh; java -Xms1g -Xmx8g -jar vf-api.jar --spring.config.location=file:/config/application.yaml"  # service:  # type: ClusterIP  # port: 9999  # external:  # enabled: true  # type: ingress #route  # host: vf-dev-dbservice  # defaultdns: vf-app #apps.okd4.okd.gomel.iba.by  # Backend History Service  # historyserv:  # deployment:  # replicas: 1  # image:  # repository: ghcr.io/ibagomel/vf-backend-history-service  # tag: latest  # pullPolicy: Always # IfNotPresent  # service:  # type: ClusterIP  # port: 9990  # external:  # enabled: true  # type: route  # host: "vf-dev-historyserv"  # defaultdns: apps.okd4.okd.gomel.iba.by  # configFile:  # postgresql:  # PG\_URL: <pg\_db\_path> # jdbc:postgresql://okd4w1.okd.gomel.iba.by:31433/example  # PG\_USER: <pg\_username> # postgres\_username  # PG\_PASS: <pg\_password> # postgres\_password |

* Create Visual Flow namespace

|  |
| --- |
| kubectl create namespace vf-test |

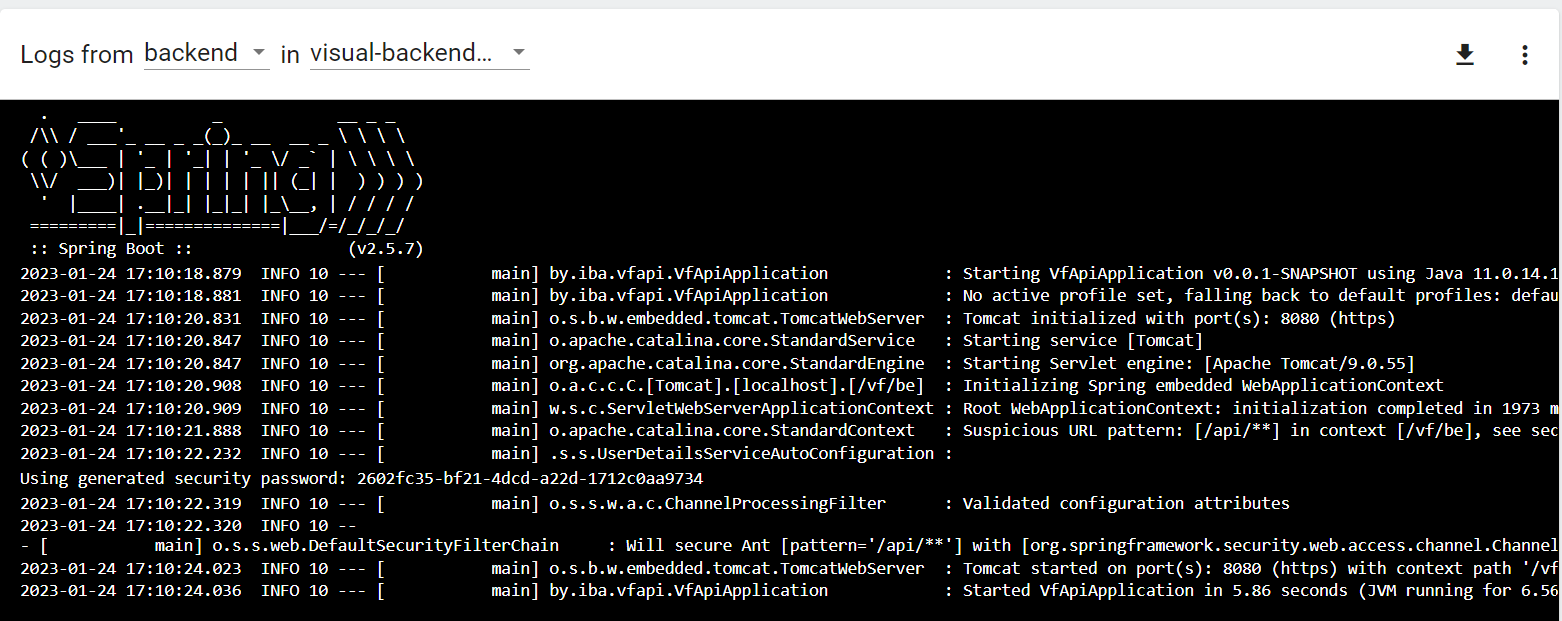
* Run helm installation command

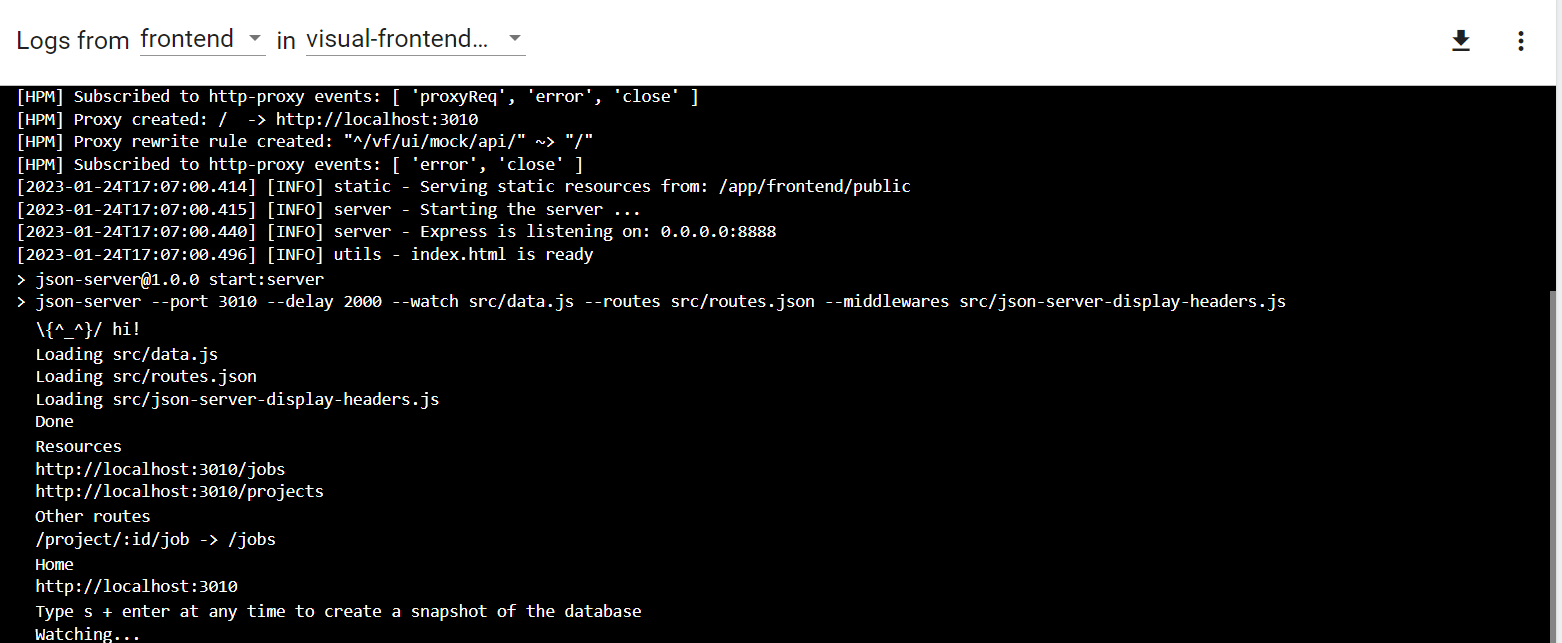
|  |
| --- |
| helm install vf-app .f values.yaml -n vf-test |

* Check pod status

Frontend

* Check pod logs

****

****

* Visit Visual Flow dashboard using link https://<local\_host\_ip>:<Node\_Port>/vf/ui/.
* Create test project. Use Visual\_Flow\_User\_Guide\_v\_<version>.pdf from [Visual-Flow](https://github.com/ibagroup-eu/Visual-Flow) repository.

# Links & Additional information

Visual Flow repository: <https://github.com/ibagroup-eu>

Visual Flow helm chart: <https://github.com/ibagroup-eu/Visual-Flow-deploy>

Visual Flow readme: <https://github.com/ibagroup-eu/Visual-Flow-deploy/blob/main/README.md>

Visual Flow values readme: <https://github.com/ibagroup-eu/Visual-Flow-deploy/blob/main/charts/vf-app/README.md>

User Guide: <https://github.com/ibagroup-eu/Visual-Flow/blob/main/Visual_Flow_User_Guide_v_1.2.0.pdf>

Contribution

<https://github.com/ibagomel/Visual-Flow/blob/main/CONTRIBUTING.md>

License

Visual Flow is an open-source software licensed under the [Apache-2.0 License](https://github.com/ibagroup-eu/Visual-Flow-deploy/blob/main/LICENSE)