Установка в microk8s

Системные требования

Приведенные ниже системные требования являются ориентировочными, минимальными. Действительные значение в зависимости от нагрузки могут варьироваться и должны выбираться исходя из задач

Операционная система	Debian-based дистрибутив Linux, desktop-версия (Предпочтительно)
Минимальные требования	Процессор: x86-x64, 2 ядра, тактовая частота 2.2 GHz и выше Жесткий диск**: SSD 25 ГБ Оперативная память: 6 ГБ
Рекомендуемые требования	Процессор: x86-x64, 4 ядра, тактовая частота 2.2 GHz и выше Жесткий диск**: SSD 40 ГБ Оперативная память: 10 ГБ

Установка Microk8s

Перед началом установки убедитесь в установке пакета snapd. Подробную информацию о snapd и его установке можно найти https://snapcraft.io/docs/installing-snapd

Для установки Microk8s необходимо выполнить следующую команду

```
sudo snap install microk8s --classic --channel=1.23/stable
```

В успешности установки и запуска можно убедится выполнив команду

```
sudo microk8s kubectl get nodes
```

Вывод этой команды должен иметь следующий вид

```
NAME                     STATUS        ROLES        AGE        VERSION
your-pc-name        Ready        <none>        24h        v1.20.13-35+d877e7a8ac536e
```

Status=Ready - свидетельствует от том что запуск выполнен успешно

После установки Microk8s необходимо установить некоторые дополнения следующей командой:

```
sudo microk8s enable storage ingress dns
```

Более подробную информацию о Microk8s и установке этого приложения можно получить по ссылке https://microk8s.io/docs/getting-started

Если вы используете ОС Windows для установки Microk8s следует обратиться к инструкции https://microk8s.io/docs/install-windows

Установка Платформы

Установка инфраструктурных сервисов

Для установки сервисов обеспечивающих инфраструктуру необходимо установить helm (версия >= 3.8.1). Для этого выполните

```
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
```

Для получения подробной информации об установке и работе helm следует обратиться к официальной документации https://helm.sh/docs/intro/install/

Если вы используете ОС Windows для установки helm следует обратиться к https://helm.sh/docs/intro/install/ (https://github.com/helm/helm/releases)

Перед продолжением дальнейшей установки следует выполнить

```
helm registry login https://harbor.effective-group.ru/share
```

и ввести полученные имя пользователя\пароль (share\Pass1234).

Также необходимо получить файл для подключения к microk8s

Выполните:

```
sudo microk8s config > ./microk8s-kubeconfig
```

Далее следует выполнить установку инфраструктурных сервисов: mongodb, redis и rabbitmq.

Установка MongoDB

Для установки mongodb в конфигурации необходимой для работы минимальной поставки Платформы следует выполинть:

```
helm install mongodb oci://harbor.effective-group.ru/share/mongodb --version 10.26.3 -f ./mongodb.values.yaml -
n mongodb --create-namespace --atomic --timeout 120s --kubeconfig=./microk8s-kubeconfig
```

Values для установки MongoDB можно использовать указанные ниже, либо внести изменения в соответствии с вашими требованиями, опираясь на официальную документацию

```yaml
architecture: standalone
persistence:
  storageClass: ""
  size: 1Gi
auth:
  enabled: true
  rootUser: mongoadmin
  rootPassword: "Pass1234"
  username: "customer"
  password: "Pass1234"
  database: "factory"
service:
  nameOverride: "mongodb-svc"
```

Для получения детальной информации об установке, работе и настройке mongodb следует обратиться к официальной документации https://www.mongodb.com/docs/

Установка Redis

Для установки redis в конфигурации необходимой для работы минимальной поставки Платформы следует выполнить:

```
helm install redis oci://harbor.effective-group.ru/share/redis --version 17.11.3 -f ./redis.values.yaml -n
redis --create-namespace --atomic --timeout 120s --kubeconfig=./microk8s-kubeconfig
```

Values для установки Redis можно использовать указанные ниже, либо внести изменения в соответствии с вашими требованиями, опираясь на официальную документацию

```yaml
architecture: standalone
auth:
  enabled: false
image:
  registry: docker.io
  repository: bitnami/redis
  tag: 7.0.5-debian-11-r14
master:
  persistence:
    enabled: true
    size: 1Gi
```

Для получения детальной информации об установке, работе и настройке redis следует обратиться к официальной документации https://redis.io/docs/

Установка RabbitMQ

Для установки rabbitmq в конфигурации необходимой для работы минимальной поставки Платформы следует выполнить:

```
helm install rabbitmq oci://harbor.effective-group.ru/share/rabbitmq --version 11.15.3 -f ./rabbitmq.values.
yaml -n rabbitmq --create-namespace --atomic --timeout 120s --kubeconfig=./microk8s-kubeconfig
```

Values для установки RabbitMQ можно использовать указанные ниже, либо внести изменения в соответствии с вашими требованиями, опираясь на официальную документацию

```
image:
  registry: docker.io
  repository: bitnami/rabbitmq
  tag: 3.8.16-debian-10-r20
  debug: false
  pullPolicy: IfNotPresent
kubeVersion:
clusterDomain: cluster.local
hostAliases: []
commonAnnotations: {}
auth:
  username: customer
  password: Pass1234
  tls:
    enabled: false
    failIfNoPeerCert: true
    sslOptionsVerify: verify_peer
    caCertificate: |-
    serverCertificate: |-
    serverKey: |-
    existingSecretFullChain: false
logs: '-'
ulimitNofiles: '65536'
memoryHighWatermark:
  enabled: false
  type: 'relative'
  value: 0.4
plugins: 'rabbitmq_management rabbitmq_peer_discovery_k8s'
extraPlugins: 'rabbitmq_auth_backend_ldap'
clustering:
  enabled: true
  addressType: hostname
  rebalance: false
  forceBoot: false
loadDefinition:
  enabled: false
terminationGracePeriodSeconds: 120
extraEnvVars: []
extraContainerPorts: []
configuration: |-
  {{- if not .Values.loadDefinition.enabled -}}
  default_user = {{ .Values.auth.username }}
  default_pass = CHANGEME
  {{- end }}
  {{- if .Values.clustering.enabled }}
  cluster_formation.peer_discovery_backend  = rabbit_peer_discovery_k8s
  cluster_formation.k8s.host = kubernetes.default.svc.{{ .Values.clusterDomain }}
  cluster_formation.node_cleanup.interval = 10
  cluster_formation.node_cleanup.only_log_warning = true
  cluster_partition_handling = autoheal
  {{- end }}
  queue_master_locator = min-masters
  loopback_users.guest = false
  {{ tpl .Values.extraConfiguration . }}
  {{- if .Values.auth.tls.enabled }}
  ssl_options.verify = {{ .Values.auth.tls.sslOptionsVerify }}
  listeners.ssl.default = {{ .Values.service.tlsPort }}
  ssl_options.fail_if_no_peer_cert = {{ .Values.auth.tls.failIfNoPeerCert }}
  ssl_options.cacertfile = /opt/bitnami/rabbitmq/certs/ca_certificate.pem
  ssl_options.certfile = /opt/bitnami/rabbitmq/certs/server_certificate.pem
  ssl_options.keyfile = /opt/bitnami/rabbitmq/certs/server_key.pem
  {{- end }}
  {{- if .Values.ldap.enabled }}
  auth_backends.1 = rabbit_auth_backend_ldap
  auth_backends.2 = internal
  {{- range $index, $server := .Values.ldap.servers }}
  auth_ldap.servers.{{ add $index 1 }} = {{ $server }}
  {{- end }}
  auth_ldap.port = {{ .Values.ldap.port }}
  auth_ldap.user_dn_pattern = {{ .Values.ldap.user_dn_pattern  }}
  {{- if .Values.ldap.tls.enabled }}
  auth_ldap.use_ssl = true
  {{- end }}
  {{- end }}
```

```
    {{- if .Values.metrics.enabled }}
    prometheus.tcp.port = 9419
    {{- end }}
    {{- if .Values.memoryHighWatermark.enabled }}
    total_memory_available_override_value = {{ include "rabbitmq.toBytes" .Values.resources.limits.memory }}
    vm_memory_high_watermark.{{ .Values.memoryHighWatermark.type }} = {{ .Values.memoryHighWatermark.value }}
    {{- end }}
extraConfiguration: |-
advancedConfiguration: |-
ldap:
  enabled: false
  servers: []
  port: '389'
  user_dn_pattern: cn=${username},dc=example,dc=org
  tls:
    enabled: false
extraVolumeMounts: []
extraVolumes: []
extraSecretsPrependReleaseName: false
extraSecrets: {}
replicaCount: 1
podManagementPolicy: OrderedReady
podLabels: {}
podAnnotations: {}
updateStrategyType: RollingUpdate
statefulsetLabels: {}
priorityClassName: ''
podAffinityPreset: ""
podAntiAffinityPreset: soft
nodeAffinityPreset:
  type: ""
  key: ""
  values: []
affinity: {}
nodeSelector: {}
tolerations: []
podSecurityContext:
  enabled: true
  fsGroup: 1001
  runAsUser: 1001
containerSecurityContext: {}
resources:
  limits: {}
  requests: {}
livenessProbe:
  enabled: true
  initialDelaySeconds: 120
  timeoutSeconds: 20
  periodSeconds: 30
  failureThreshold: 6
  successThreshold: 1
readinessProbe:
  enabled: true
  initialDelaySeconds: 10
  timeoutSeconds: 20
  periodSeconds: 30
  failureThreshold: 3
  successThreshold: 1
customLivenessProbe: {}
customReadinessProbe: {}
customStartupProbe: {}
serviceAccount:
  create: true
rbac:
  create: true
persistence:
  enabled: true
  storageClass: ""
  selector: {}
  accessMode: ReadWriteOnce
  size: 1Gi
  volumes:
pdb:
  create: false
  minAvailable: 1
networkPolicy:
  enabled: false
  allowExternal: true
service:
  type: ClusterIP
  port: 5672
  portName: amqp
  tlsPort: 5671
  tlsPortName: amqp-ssl
```

```
    distPort: 25672
    distPortName: dist
    managerPortEnabled: true
    managerPort: 15672
    managerPortName: http-stats
    metricsPort: 9419
    metricsPortName: metrics
    epmdPortName: epmd
    extraPorts: []
    externalTrafficPolicy: Cluster
    labels: {}
    annotations: {}
    annotationsHeadless: {}
  ingress:
    enabled: false
    path: /
    pathType: ImplementationSpecific
    hostname: rabbitmq.local
    annotations: {}
    tls: false
    certManager: false
    selfSigned: false
    extraHosts: []
    extraTls: []
    secrets: []
  metrics:
    enabled: false
    plugins: 'rabbitmq_prometheus'
    podAnnotations:
      prometheus.io/scrape: 'true'
      prometheus.io/port: '{{ .Values.service.metricsPort }}'
    serviceMonitor:
      enabled: false
      interval: 30s
      honorLabels: false
      additionalLabels: {}
      targetLabels:
      podTargetLabels:
    prometheusRule:
      enabled: false
      additionalLabels: {}
      namespace: ''
      rules: []
  volumePermissions:
    enabled: false
    image:
      registry: docker.io
      repository: bitnami/bitnami-shell
      tag: 10-debian-10-r93
      pullPolicy: Always
      pullSecrets: []
    resources:
      limits: {}
      requests: {}
```

Также для корректной работы rabbitmq следует выполнить настройку используя следующую команду:

```
sudo microk8s kubectl exec -it svc/rabbitmq -n rabbitmq -- bash -c "rabbitmqctl add_vhost factory &&
rabbitmqctl set_permissions -p factory customer \".*\" \".*\" \".*\""
```

Для получения детальной информации об установке, работе и настройке rabbitmq следует обратиться к официальной документации https://www.rabbitmq.com/documentation.html

Установка платформы

Для установки платформы следует выполнить следующую команду:

```
helm install factory-deploytoenv oci://harbor.effective-group.ru/share/deploytoenv --version 0.0.3 -f .
/deploytoenv.values.yaml -n factory --create-namespace --atomic --timeout 60s --kubeconfig=./microk8s-kubeconfig
```

где deploytoenv.values.yaml имеет следующее содержание:

```
serviceName: "deploytoenv"
serviceVersion: ""

image:
  registry: "harbor.effective-group.ru"
  repository: "share"
  imageName: "deploytoenv"
  tag: "0.1.3"
  credentials:
    username: "share"
    password: "Pass1234"
```

Теперь перейдем непосредственно к установке Платформы. Для этого выполните команду:

```
helm install factory oci://harbor.effective-group.ru/share/factory --version 12.13.6 -f ./factory.values.yaml -
n factory --create-namespace --atomic --timeout 600s --kubeconfig=./microk8s-kubeconfig
```

где factory.values.yaml имеет следующее содержимое, но может быть изменен в соответствии с вашими требованиями:

```
global:
  domain: "factory.eftech.online"
  envName: "factory"

  image:
    registry: "harbor.effective-group.ru"
    repository: "share"
    credentials:
      username: "share"
      password: "Pass1234"

  serviceAccount:
    create: true
    name: factory

  dataService: "http://data"
  workflowService: "http://workflow"
  workflowV2Service: "http://workflow"
  fileService: "http://file"
  cryptoService: ""

  hpa:
    enabled: false
    minReplicas: 1
    maxReplicas: 2
    targetMemoryUtilizationPercentage: 75
    targetCPUUtilizationPercentage: 80

  captchaSiteKey: ""

  ingress:
    clusterIssuer: letsencrypt-prod
    whitelist: false
    httpsRedirect: false
    robotsTxt: false
    protocol: "http"

  replicaCount: 1

  tracing:
    enabled: "false"
    jaegerAgentHost: jaeger
    jaegerAgentPort: 6832
    jaegerAgentPortGo: 6831
    tracingUrl: "https://factory.eftech.online/tracing/search?service=api-proxy"

  timeout: 120000
  sessionExpiration: 3600
  singleSession: false
  nodeEnv: production
  demoFunctions: true
  withPublicStats: true

  mongodb:
    dsn: "mongodb://customer:Pass1234@mongodb.mongodb:27017/factory"
    dbName: "factory"
    port: "27017"

  rabbitmq:
    host: "rabbitmq.rabbitmq"
    port: "5672"
    user: "customer"
```

```
    password: "Pass1234"
    vhost: "factory"

  redis:
    mode: single
    sentinel:
      addresses: ""
    port: 6379
    host: redis-master.redis

  configEnvs:
    ENV_NAME: "factory"

  configurator:
    serviceVersion: "12.13.57-local"
    domain: "configurator.factory.eftech.online"
    image:
      repository: "share/configurator"
api-proxy:
  serviceVersion: "12.13.53"
  resources:
    requests:
      cpu: 150m
      memory: 200Mi

configurator:
  configurator-api:
    resources:
      requests:
        cpu: 200m
        memory: 200Mi

    configEnvs:
      APP_MODE: "extended"
      RESOURCES_PATH: "/resources"
  configurator-frontend:
    resources:
      requests:
        cpu: 100m
        memory: 50Mi
    configEnvs:
      APP_MODE: "extended"
  configurator-resourcesmaster:
    resources:
      requests:
        memory: 400Mi
        cpu: 200m
    configEnvs:
      RESOURCES_PATH: "/resources"
event-logger:
  serviceVersion: "12.13.1"
  resources:
    requests:
      cpu: 100m
      memory: 100Mi

file:
  serviceVersion: "12.13.2"
  filesQuota: 1Gi
  storageClass: microk8s-hostpath
  defaultStorage: local
  s3Enabled: false
  s3RootPath: "/factory"
  resources:
    s3:
      limits:
        cpu: 100m
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 200Mi
  configEnvs:
    BASE_PATH: "/upload/"

frontend:
  serviceVersion: "12.13.46"
  wsApi: "wss://factory.eftech.online/ws"
  resources:
    frontend:
      requests:
        cpu: 50m
        memory: 100Mi
      limits:
        cpu: 50m
```

```
        memory: 100Mi
    sidecar:
      requests:
        cpu: 200m
        memory: 300Mi
      limits:
        cpu: 400m
        memory: 300Mi

cache:
  serviceVersion: "12.13.3"
  image:
    imageName: cache-service
  resources:
    requests:
      cpu: 50m
      memory: 50Mi

data:
  serviceVersion: "12.13.40"
  resources:
    requests:
      cpu: 250m
      memory: 700Mi
    limits:
      cpu: 2
      memory: 1400Mi

document:
  serviceVersion: "12.13.28"
  resources:
    requests:
      cpu: 50m
      memory: 200Mi
    limits:
      cpu: 1
      memory: 400Mi

gateway:
  image:
    repository: "share/integration"
  serviceVersion: "12.13.36"
  resources:
    requests:
      cpu: 100m
      memory: 100Mi

  configEnvs:
    WORKFLOW_SERVICE_URL: "http://workflow"
    ENV_NAME: "factory"

migrate:
  serviceVersion: "12.13.3"
  image:
    repository: "share/migrate"
    imageName: "12.13.x"
    tag: 6ac9bb07-127283
notification:
  serviceVersion: "12.13.4"
  image:
    imageName: notification-service
  rabbitmqVhost: "factory"
  smtpPort: 1025
  smtpServer: mailhog.eftech.local
  fromEmail: no-reply@effective-group.ru
  fromName: " (15.0.x/release)"
  resources:
    sender:
      requests:
        memory: 100Mi
        cpu: 20m
    handler:
      requests:
        memory: 50Mi
        cpu: 20m
    delaySender:
      requests:
        memory: 20Mi
        cpu: 10m
    cancel:
      requests:
        memory: 20Mi
        cpu: 10m
```

```
  settings:
    serviceVersion: "12.13.34"
    image:
      imageName: settings-service
    resources:
      requests:
        cpu: 50m
        memory: 150Mi

  workflow:
    serviceVersion: "12.13.28"
    resources:
      requests:
        cpu: 50m
        memory: 150Mi

    configEnvs:
      SERVICE_NAME: workflow
      SERVICE_VERSION: v2

  log-api:
    serviceVersion: "12.13.6"
    resources:
      requests:
        cpu: 50m
        memory: 100Mi

    configEnvs:
      LOG_LEVEL: info
```

После установки Платформы также следует выполнить команды:

```
sudo microk8s kubectl scale --replicas=0 statefulset file -n factory
```

и

```
sudo microk8s kubectl annotate --overwrite ingress --all -n factory kubernetes.io/ingress.class=public
```

Заключительным шагом установки нужно добавить в файл /etc/hosts следующие записи (если вы используете ос Windows изменения следует внести в C:\Windows\System32\drivers\etc\hosts):

```
127.0.0.1       configurator.factory.eftech.online
127.0.0.1       factory.eftech.online
```

Теперь доступ к Платформе вы можете получить открыв в браузере http://factory.eftech.online