Инструкция по установке экземпляра программного обеспечения «Росдомофон» и его функциональных модулей «Личный кабинет», «Умный домофон», «Видеонаблюдение», «Калитка», «Мобильное приложение»

***Содержит информацию, необходимую для установки** программы «Росдомофон» и ее функциональных модулей*

Содержание

Настоящий документ представляет собой инструкцию по установке программы для ЭВМ «Росдомофон» (далее по тексту – программа) и ее функциональных модулей.

Программа для ЭВМ «Росдомофон» представляет собой программный комплекс, имеет модульную структуру и включает в себя следующие функциональные модули: «Личный кабинет», «Умный домофон», «Видеонаблюдение», «Калитка», «Мобильное приложение».

Программа представляет собой облачную платформу и предоставляет инструменты дистанционного управления устройствами на придомовой территории, в жилых и не жилых помещениях, совершение вызова в приложение абонента, а также просмотр видео в режиме реального времени и архивных записей видеонаблюдения.

1. Установка системы виртуализации

Для создания системы виртуализации необходимо установить **Proxmox VE** минимум на 3 сервера (это количество необходимо для развертывания инфраструктуры с достаточным резервированием).

Рекомендуемая конфигурация для серверов, на которые будет производиться установка:

- Intel Xeon Silver 4210, 256 Гб оперативной памяти;
- 2 HDD 4 Тб и 2 SSD 1 Тб (под виртуальные машины);
- 1 SSD 128 Гб (под систему и образы дисков).

1.1. Установка Proxmox

Образ диска с готовой системой Proxmox VE последней версии: https://www.proxmox.com/en/downloads/category/iso-images-pve2.2.

Установка производится со стандартными настройками на диск 128Гб, изменить необходимо только часовой пояс, пароль суперпользователя и сетевые настройки.

1.2. Настройка

Настройку удобнее производить через web-интерфейс, используя IP-адрес, указанный при установке, и порт 8006.

1.2.1. Объединение серверов в кластер

Следуя инструкции https://pve.proxmox.com/wiki/Cluster_Manage, необходимо на одном из серверов создать кластер, выбрав **Create Cluster**. Для удобства присоединения остальных машин к кластеру рекомендуется **скопировать данные присоединения** (Join Information).

Затем с помощью **Join Cluster** нужно последовательно присоединить к кластеру остальные серверы (данные присоединения вставляются в соответствующее поле).

1.2.2. Организация общего хранилища ceph

Необходимо создать 2 виртуальных раздела: один на SSD, второй на HDD.

Инструкция: https://pve.proxmox.com/wiki/Deploy_Hyper-Converged_Ceph_Cluster.

1.2.3. Создание пользователя для Terraform

Чтобы **создать пользователя для Terraform**, нужно выполнить на любом из серверов кластера следующие команды:

Создание пользователя Proxmox для Terraform:

```
pveum role add TerraformRole -privs "VM.Allocate VM.Clone VM.Config.CDROM
VM.Config.CPU VM.Config.Cloudinit VM.Config.Disk VM.Config.HWType
VM.Config.Memory VM.Config.Network VM.Config.Options VM.Monitor VM.Audit
VM.PowerMgmt Datastore.AllocateSpace Datastore.Audit Pool.Allocate Pool.Audit"

pveum user add terraform-pve@pve --password <password>

pveum aclmod / -user terraform-pve@pve -role TerraformProv
```

2. Создание виртуальных машин

Для создания виртуальных машин используется **Terraform** и его модуль **Telmate/proxmox** для управления кластером виртуализации.

Файлы, необходимые для **развертывания виртуальных машин**, расположены в репозитории https://gitlab.rosdomofon.com/next_level/terraform-provider-proxmox.

Описание виртуальных машин находится в файле https://gitlab.rosdomofon.com/next_level/terraform-provider-proxmox/-/blob/master/terraform/main.tf.

Для **master-** и **worker-**узлов предпочтительнее использовать приватные IP-адреса, а **ingress** необходим ещё и публичный адрес (для принятия соединений извне кластера).

Пример конфигурации master-узла:

```
resource "proxmox_vm_qemu" "dev-master-01" {
    name = "dev-master-01"
    desc = "K8S dev master-01"
    onboot = true
    vmid = 200
    target_node = "pvetest1"
    pool = "pool0"
    clone = "ubuntu20-cloud-init"
    boot = "c"
    bootdisk = "scsi0"
    os_type = "cloud-init"
    cores = 2
    sockets = 2
    memory = 8192
    scsihw = "virtio-scsi-single"
    ciuser = "ansibleRD"
    sshkeys = var.ansible_key
    disk {
        size = "30G"
        type = "scsi"
        storage = "local-lvm"
        iothread = 1
        discard = "on"
    }
    network {
        model = "virtio"
        bridge = "vmbr1"
    }
    ipconfig0 = "ip=10.20.10.150/24,gw=10.20.10.1"
}
```

Пример конфигурации worker-узла:

```
resource "proxmox_vm_qemu" "dev-worker-01" {
    name = "dev-worker-01"
    desc = "K8S dev worker-01"
    onboot = true
    vmid = 203
    target_node = "pvetest1"
    pool = "pool0"
    clone = "ubuntu20-cloud-init"
    boot = "c"
    bootdisk = "scsi0"
    os_type = "cloud-init"
    cores = 4
    sockets = 2
    memory = 32768
    scsihw = "virtio-scsi-single"
    ciuser = "ansibleRD"
    sshkeys = var.ansible_key
    disk {
        size = "30G"
        type = "scsi"
        storage = "local-lvm"
        iothread = 1
        discard = "on"
    }
    disk {
        size = "100G"
        type = "scsi"
        storage = "VMhdd"
        iothread = 1
        discard = "on"
    }
    network {
        model = "virtio"
        bridge = "vmbr1"
    }
    ipconfig0 = "ip=10.20.10.153/24,gw=10.20.10.1"
}
```

Пример конфигурации ingress-узла:

```
resource "proxmox_vm_qemu" "dev-ingress-01" {
    name = "dev-ingress-01"
    desc = "K8S dev ingress 01"
    onboot = true
    vmid = 206

    target_node = "pvetest3"
    pool = "pool0"
    clone = "ubuntu20-cloud-init"
    boot = "c"
    bootdisk = "scsi0"
    os_type = "cloud-init"
    cores = 2
    sockets = 2
    memory = 4096
    scsihw = "virtio-scsi-single"
    ciuser = "ansibleRD"
    sshkeys = var.ansible_key
    disk {
        size = "30G"
        type = "scsi"
        storage = "VMhdd"
        iothread = 1
        discard = "on"
    }
    network {
        model = "virtio"
        bridge = "vmbr0"
    }
    network {
        model = "virtio"
        bridge = "vmbr1"
    }
    ipconfig0 = "ip=185.111.219.180/24,gw=185.111.219.1"
    ipconfig1 = "ip=10.20.10.156/24"
}
```

За применение файла состояния (Terraform state) в кластере отвечают механизмы **CI/CD Gitlab**, логика работы которых описана в соответствующих файлах репозитория:

1) **На первом шаге (build, сборка)** формируется конфигурация для применения — план действий: сравнивается текущее состояние и описание в репозитории. Рекомендуется проверять вручную, что изменяются/добавляются/удаляются только нужные машины. Лог CI должен содержать строку с описанием, сколько машин запланировано к созданию, изменению и удалению, например "`Plan: 6 to add, 2 to change, 0 to destroy`", и заканчиваться записью "`Job succeeded`".
2) **На втором шаге (deploy, развертывание)** происходит применение сформированного на первом шаге плана (т.е., выполняется создание, изменение и удаление машин).

Пример успешного окончания deploy:

```
Apply complete! Resources: 6 added, 2 changed, 0 destroyed.
Saving cache for successful job
Creating cache terraform-protected...
terraform/.terraform/: found 11 matching files and directories
No URL provided, cache will not be uploaded to shared cache server. Cache will be
stored only locally.
Created cache
Cleaning up project directory and file based variables
Job succeeded
```

3. Создание кластера Kubernetes

Для создания кластера Kubernetes используется **Kubespray**, настройки которого находятся в репозитории https://gitlab.rosdomofon.com/next_level/kubespray/-/tree/rd-dev-from-2.20/inventory/rd-dev. Основным является файл hosts.yaml с добавленными в него IP-адресами созданных виртуальных машин.

IP-адреса тестового кластера:

```
hosts:
    dev-master-01:  { ansible_host: 10.20.10.150 }
    dev-master-02:  { ansible_host: 10.20.10.151 }
    dev-master-03:  { ansible_host: 10.20.10.152 }
    dev-worker-01:  { ansible_host: 10.20.10.153 }
    dev-worker-02:  { ansible_host: 10.20.10.154 }
    dev-worker-03:  { ansible_host: 10.20.10.155 }
    dev-ingress-01: { ansible_host: 185.111.219.180 }
    dev-rdva-01:    { ansible_host: 185.111.219.181 }
    dev-rdva-02:    { ansible_host: 185.111.219.182 }
    dev-rdva-03:    { ansible_host: 185.111.219.183 }
    dev-rdva-04:    { ansible_host: 185.111.219.184 }
```

Развертывание реализовано средствами Gitlab CI/CD.

Пример успешного завершения процесса:

```
PLAY RECAP *********************************************************************
dev-ingress-01             : ok=490  changed=4    unreachable=0    failed=0
skipped=760  rescued=0    ignored=2
dev-master-01              : ok=722  changed=15   unreachable=0    failed=0
skipped=1243 rescued=0    ignored=2
dev-master-02              : ok=617  changed=8    unreachable=0    failed=0
skipped=1096 rescued=0    ignored=2
dev-master-03              : ok=619  changed=8    unreachable=0    failed=0
skipped=1094 rescued=0    ignored=2
dev-rdva-01                : ok=493  changed=21   unreachable=0    failed=0
skipped=760  rescued=0    ignored=2
dev-rdva-02                : ok=493  changed=21   unreachable=0    failed=0
skipped=760  rescued=0    ignored=2
dev-rdva-03                : ok=493  changed=20   unreachable=0    failed=0
skipped=760  rescued=0    ignored=2
dev-rdva-04                : ok=493  changed=21   unreachable=0    failed=0
skipped=760  rescued=0    ignored=2
dev-worker-01              : ok=488  changed=4    unreachable=0    failed=0
skipped=765  rescued=0    ignored=2
dev-worker-02              : ok=488  changed=4    unreachable=0    failed=0
skipped=762  rescued=0    ignored=2
dev-worker-03              : ok=488  changed=4    unreachable=0    failed=0
skipped=762  rescued=0    ignored=2
localhost                  : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
Friday 23 December 2022  10:23:49 +0000 (0:00:00.443)       0:32:13.965 *******
```

```
===============================================================================
download : download_file | Validate mirrors ---------------------------- 74.27s
container-engine/validate-container-engine : Populate service facts ---- 50.58s
kubernetes/preinstall : Update package management cache (APT) ---------- 36.53s
kubernetes/kubeadm : Restart all kube-proxy pods to ensure that they load the new
configmap -- 24.83s
container-engine/containerd : download_file | Download item ------------ 24.35s
etcd : Gen_certs | Write etcd member/admin and kube_control_plane clinet certs to
other etcd nodes -- 19.83s
download : download | Download files / images -------------------------- 17.45s
container-engine/containerd : containerd | Unpack containerd archive --- 17.39s
container-engine/nerdctl : extract_file | Unpacking archive ------------ 16.42s
container-engine/crictl : extract_file | Unpacking archive ------------- 16.34s
container-engine/runc : download_file | Download item ------------------ 15.46s
container-engine/crictl : download_file | Download item ---------------- 14.96s
container-engine/nerdctl : download_file | Download item --------------- 14.81s
container-engine/runc : download_file | Validate mirrors --------------- 11.37s
container-engine/crictl : download_file | Validate mirrors ------------- 11.23s
container-engine/containerd : download_file | Validate mirrors --------- 11.01s
container-engine/nerdctl : download_file | Validate mirrors ------------ 10.99s
download : download_file | Download item ------------------------------- 9.84s
download : download | Download files / images -------------------------- 9.77s
download : download | Download files / images -------------------------- 9.70s
Cleaning up project directory and file based variables
Job succeeded
```

4. ArgoCD

За развертывание сервисов в кластер отвечает **ArgoCD,** который необходимо установить с помощью репозитория https://gitlab.rosdomofon.com/next_level/argocd, дополнив установку ручными действиями, описанными в файле readme.md.

Установка начинается с **создания аккаунта**, с помощью которого будут выполняться действия в Kubernetes. Для этого нужно выполнить команду `kubectl apply -f sa-secret-clusterrolebinding.yaml`.

sa-secret-clusterrolebinding.yaml

```
---
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: gitlab
  namespace: kube-system
  annotations:
    kubernetes.io/service-account.name: "gitlab"
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: gitlab
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: gitlab-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: gitlab
    namespace: kube-system
```

При создании пользователя (ServiceAccount) Kubernetes сгенерирует токен, который нужно применить при создании конфигурационного файла для доступа в кластер Kubernetes.

Для создания **конфигурационного файла** используется команда `kubectl -n kube-system get secret gitlab -o jsonpath='{.data.token}' | base64 --decode; echo`.

Также с помощью команды нужно получить сертификат certificate-authority-data: `kubectl -n kube-system get secret gitlab -o jsonpath='{.data.ca.crt}'; echo`.

Полученный конфигурационный файл будет использоваться для развертывания ArgoCD, поэтому его нужно сохранить в переменную типа File с именем $DEV_KUBECONFIG (в настройках проекта/группы Settings → CI/CD → Variables).

Переменная KUBECONFIG:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0***_ИЗ_ПРЕДЫДУЩЕГО_ШАГА_***
    server: https://10.20.10.150:6443
  name: test-cluster.local
contexts:
- context:
    cluster: test-cluster.local
    user: gitlab
  name: default
current-context: default
kind: Config
preferences: {}
users:
- name: gitlab
  user:
    token: ey***_ИЗ_ПРЕДЫДУЩЕГО_ШАГА_***
```

Далее необходимо произвести развёртывание средствами **Gitlab CI/CD**.

В процессе развертывания, помимо установки компонентов ArgoCD, происходит добавление репозитория с учетными данными для доступа к нему.

```
kubectl --kubeconfig=${DEV_KUBECONFIG} -n argocd create secret generic
gitlab-all-repos --type=git --from-literal=url=${REPOURL}
--from-literal=password=${GITLAB_ARGOCD_PASS}
--from-literal=username=${GITLAB_ARGOCD_USR} --dry-run -o yaml | kubectl
--kubeconfig=${DEV_KUBECONFIG} apply -f -  && kubectl
--kubeconfig=${DEV_KUBECONFIG} -n argocd label secret gitlab-all-repos
argocd.argoproj.io/secret-type=repo-creds --overwrite
```

${GITLAB_ARGOCD_USR} и ${GITLAB_ARGOCD_PASS} — переменные, хранящие имя пользователя Gitlab и сгенерированный для него Access Token.

После развертывания нужно сохранить в переменную (Settings → CI/CD → Variables) $ARGOCDPASS секрет argocd-initial-admin-secret из namespace argocd (`kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 --decode ; echo`).

Также необходимо добавить переменную $ARGOCD_USER="admin".

Обе эти переменные будут полезны для дальнейших развертываний из Gitlab в ArgoCD во избежание добавления приложения вручную через web-интерфейс.

5. Добавление сервисов

5.1. Добавление сервисов в ArgoCD

Необходимо добавить в ArgoCD инфраструктурные сервисы из репозитория https://gitlab.rosdomofon.com/next_level/infra-apps/app-of-apps.

Репозиторий можно добавить как с помощью Gitlab CI/CD, так и через web-интерфейс. Независимо от выбранного способа, результатом синхронизации будут готовые инфраструктурные сервисы в кластере.

Сервисы описаны файлами в директории templates (файлы самих сервисов находятся в репозитории на уровень выше). ArgoCD создает приложения, основываясь на этих описаниях.

Пример успешного завершения deploy:

```
Message:            successfully synced (all tasks run)
GROUP        KIND         NAMESPACE  NAME                       STATUS     HEALTH
HOOK  MESSAGE
             Namespace    argocd     infra-dev                  Succeeded  Synced
namespace/infra-dev created
             Namespace    argocd     logging                    Succeeded  Synced
namespace/logging created
             Namespace    argocd     monitoring                 Succeeded  Synced
namespace/monitoring created
argoproj.io  Application  argocd     adminer                    Synced
application.argoproj.io/adminer created
argoproj.io  Application  argocd     cert-manager               Synced
application.argoproj.io/cert-manager created
argoproj.io  Application  argocd     loki-simple-scalable       Synced
application.argoproj.io/loki-simple-scalable created
argoproj.io  Application  argocd     config-syncer              Synced
application.argoproj.io/config-syncer created
argoproj.io  Application  argocd     sealed-secrets             Synced
application.argoproj.io/sealed-secrets created
argoproj.io  Application  argocd     zalando-cluster-manifests  Synced
application.argoproj.io/zalando-cluster-manifests created
argoproj.io  Application  argocd     promtail                   Synced
application.argoproj.io/promtail created
argoproj.io  Application  argocd     cert-manager-manifests     Synced
application.argoproj.io/cert-manager-manifests created
argoproj.io  Application  argocd     zalando-postgres-operator  Synced
application.argoproj.io/zalando-postgres-operator created
argoproj.io  Application  argocd     kube-prometheus-stack      Synced
application.argoproj.io/kube-prometheus-stack created
argoproj.io  Application  argocd     strimzi-kafka-operator     Synced
application.argoproj.io/strimzi-kafka-operator created
argoproj.io  Application  argocd     related-resources          Synced
application.argoproj.io/related-resources created
argoproj.io  Application  argocd     strimzi-cluster-manifests  Synced
application.argoproj.io/strimzi-cluster-manifests created
             Namespace                infra-dev                  Synced
```

```
            Namespace               logging                   Synced
            Namespace               monitoring                Synced
Cleaning up project directory and file based variables
Job succeeded
```

5.2. Развертывание пользовательских сервисов

Развертывание пользовательских сервисов выполняется из репозитория https://gitlab.rosdomofon.com/next_level/rd_apps_charts/app-of-apps.

Логика его работы аналогична описанному в п-те 5.1: каждый шаблон (template) представляет собой ссылку на репозиторий с приложением, которое автоматически разворачивается и запускается средствами ArgoCD.

6. Контакты

Наименование организации: ООО «Дом-ИТ»

ИНН/ОГРН: 9709014190/5177746111923

Телефон технической поддержки: 8(800)550-08-23

Адрес электронной почты технической поддержки: support@rosdomofon.com