EKS easy-going operations and management via eksctl

Александр Изюмов, AWS Sr Solutions architect

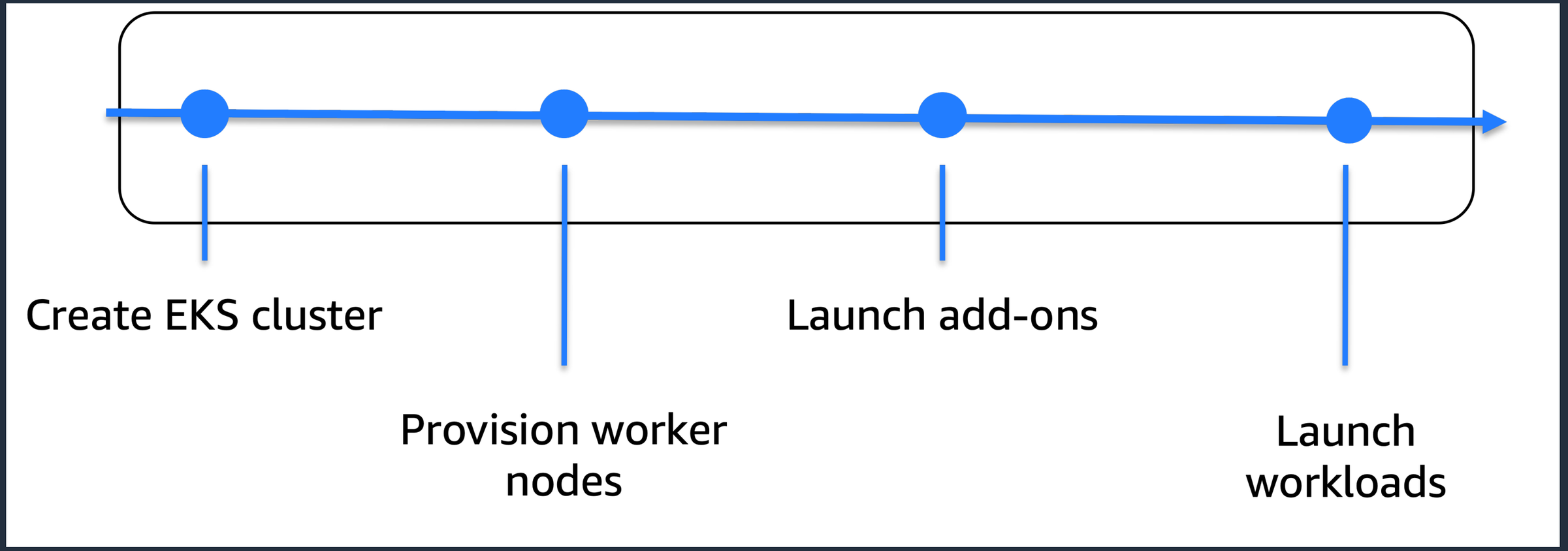
July 2020

Список тем на сегодня

- EKS provision steps & EKS communication diagram
- Часть 1 – зачем нам eksctl
- Часть 2 – базовые штуки для eksctl
- Часть 3 – важные штуки для eksctl
- Часть 4 – продакшн штуки для eksctl
- Часть 5 – остальное и новое ☺

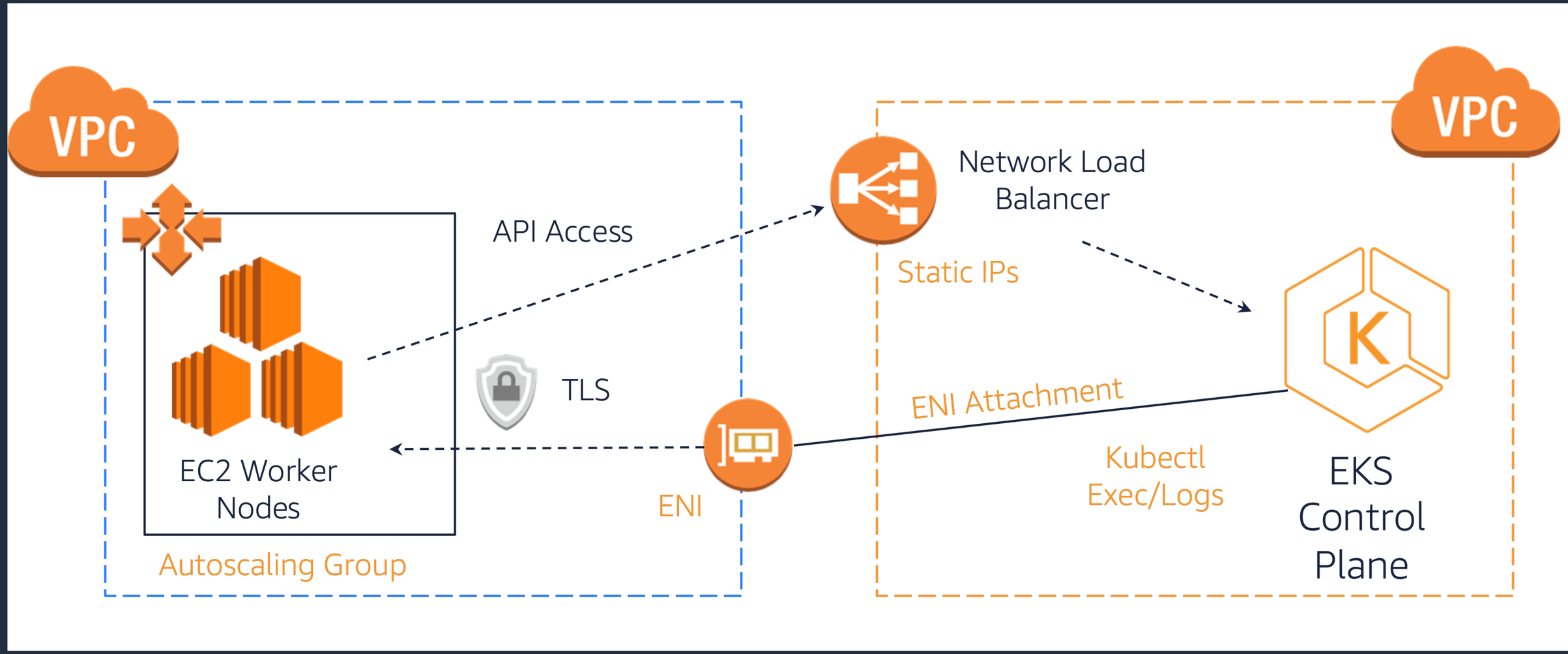
aws

Что происходит во время создания EKS кластера



aws

CONTROL PLANE AND WORKER NODE COMMUNICATION



ENI = Elastic Network Interface

aws

Cloud9 env + eksworkshop pre-setup

Cloud9 – in-browser IDE, git-integration, code-formatting, terminals, etc.

Для правильной работы с Amazon EKS кластерами – нужно настроить окружение.

https://www.eksworkshop.com/020_prerequisites/workspace/

- Create IAM Role (for EKS management) and attach to Cloud9 instance
- Update IAM setting in Cloud9
    - set “**AWS managed temporary credentials**” = OFF
    - EKS RBAC должен ссылаться на определенные, а не Temp creds

Давайте посмотрим вживую…

aws

Часть 1 – зачем нам eksctl

aws

Какие задачи по управлению EKS хотим решить?

1. Упростить/автоматизировать процедуру апгрейда EKS
2. EKS IaC (~98%) via cluster_config.yaml
    - VPC/Networking/Public-Private subnets
    - Production-grade setup
3. Setup IAM (Identity & Access Management)/Security for PODs
    - ServiceRoles для WorkerNodes
    - RBAC/IAM
4. Централизованное управление WorkerNodes разных вариантов:
    - SPOT / EC2 / Managed nodes / Fargate

aws

Часть 2 – базовые штуки для eksctl

aws

О каких настройках по-умолчанию нужно знать и какие - применять

1. Tags – нужны всегда
    - --tags environment=staging (через cli, или в yaml)
2. Выбрать нужный регион
    - --region=eu-west-2 (через cli, или в yaml)
3. Используем cluster_config.yaml (IaC для всей конфигурации)
    - eksctl create cluster -f cluster_config.yaml

aws

Что еще можно настроить?

1. Не нужен kube config после создания кластера?
    - `--write-kubeconfig=false`
2. Опять стал нужен kube config? Или потеряли, или поломался?
    - `eksctl utils write-kubeconfig --cluster=<cluster_name>`
    - `aws eks --region eu-west-2 update-kubeconfig --name <cluster_name>`
3. Хотим создать инфраструктуру (VPC/EKS), а NodeGroups – позже?
    - `--without-nodegroup`
4. Позже наступило? Добавляем:
    - `eksctl create nodegroup -f cluster_config.yaml`

aws

Custom VPC CIDR

Default VPC CIDR=192.168.0.0/16

Пример custom vpc-cidr через eksctl cli:

```
eksctl create cluster --region=eu-west-2 --vpc-cidr 10.100.0.0/24
```

Или через **cluster_config.yaml**:

```yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: cluster-2
  region: eu-north-1

vpc:
  cidr: 10.10.0.0/16
  autoAllocateIPv6: true
```


aws

Demo – review basic cluster

Yaml, EKS console, EC2 console, Container Insights, CloudWatch EKS logs

aws

Управление и настройка NodeGroups

Важно! NodeGroups == immutable, основаны на EC2 ASG (Auto-Scaling Groups) – логическая группа ресурсов EC2 с общим управлением ресурсами и масштабированием.

Можно менять только параметры масштрабирования: *количество нод*.

Чтобы изменить парамены NodeGroups, нужно:

-> создать новую NodeGroups

-> удалить старую NodeGroups Опасно? Нет!

И все это храним и применяем через cluster_config.yaml (IaC)

aws

Какие виды NodeGroups бывают

1. ”Regular” EC2
2. SPOT
3. Managed NodeGroups (aka ASG EC2 managed by EKS Control Plane)
4. Fargate for EKS

Можно миксовать все 4 типа NodeGroups!

Можно добавлять TAGS, можно скейлить.

aws

Запускаем создание Node Groups

aws

Перерыв на Q&A (short)

Managed NodeGroups vs “regular” – un/supported features

Tags (managedNodeGroups[*].tags) in managed nodegroups apply to the EKS Nodegroup resource and do not propagate to ASG

iam.instanceProfileARN is ***not supported***

The amiFamily field supports **only** AmazonLinux2

instancesDistribution field is ***not supported***

volumeSize is the only field supported for configuring volumes

Control over the node bootstrapping process and customization of the kubelet ***not supported fields***: maxPodsPerNode, taints, preBootstrapCommands, overrideBootstrapCommand, clusterDNS and kubeletExtraConfig.

См. полный список: https://eksctl.io/usage/eks-managed-nodes/#feature-parity-with-unmanaged-nodegroups

aws

SPOT-based NodeGroups

SPOT parameters:

OnDemandBaseCapacity: default=0

onDemandPercentageAboveBaseCapacity : default=100

An update to this setting means a gradual replacement of instances to maintain the specified number of On-Demand Instances for your base capacity. When replacing instances, Amazon EC2 Auto Scaling launches new instances before terminating the old ones.

spotAllocationStrategy: default=lowest-price

maxPrice: The maximum price per unit hour that you are willing to pay for a Spot Instance.

instanceTypes: t3.small, m5.large, etc.

aws

IAM policies for NodeGroups

IAM policies для NodeGroups (на картинке): EC2, ManagedNodeGroups, SPOT

attachPolicyARNs – ваши custom Policies
в дополнение к списку:

```yaml
nodeGroups:
  - name: my-special-nodegroup
    iam:
      attachPolicyARNs:
        - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
        - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
        - arn:aws:iam::aws:policy/ElasticLoadBalancingFullAccess
        - arn:aws:iam::1111111111:policy/kube2iam
      withAddonPolicies:
        autoScaler: true
        imageBuilder: true
```

```yaml
nodeGroups:
  - name: ng-1
    instanceType: m5.xlarge
    desiredCapacity: 1
    iam:
      withAddonPolicies:
        imageBuilder: true
        autoScaler: true
        externalDNS: true
        certManager: true
        appMesh: true
        ebs: true
        fsx: true
        efs: true
        albIngress: true
        xRay: true
        cloudWatch: true
```


aws

Fargate support via Fargate profile

Что такое Fargate:

On-demand эластичный пул compute ресурсов (CPU, RAM) – не нужно думать о серверах, их типах. Вы просто задаёте объём нужных ресурсов в рамках PODs и всё.

(*) Fargate adds 256 MB to each pod's memory reservation for the required Kubernetes components (kubelet, kube-proxy, and containerd).

Каждый POD на Fargate имеет уровень изоляции, и не разделяет kernel, CPU, RAM, или ENI с другими PODs.

Для запуска PODs на Fargate – нужно определить **Fargate profile**.

Fargate profile – описывает, какие PODs будут запускаться на Fargate.

aws

Fargate support via Fargate profile

Fargate profile включает в себя:

- Pod execution role
    - kubelet auth при запуске
    - IAM r/o Amazon ECR (Container registry)
- Subnet (private only)
- Selectors (namespace, labels (optional)) – до 5 штук

При создании кластера –-fargate (default profile): ***default*** and the ***kube-system*** namespaces

Создаем Fargate профиль с именем **fp-dev:**

```
eksctl create fargateprofile --namespace dev --cluster fargate-cluster --name fp-dev
```


aws

Fargate profile via cluster_config.yaml

Тоже самое из yaml config:

eksctl create fargateprofile -f fargate-example-cluster.yaml

```yaml
fargateProfiles:
  - name: fp-default
    selectors:
      # All workloads in the "default" Kubernetes namespace will be
      # scheduled onto Fargate:
      - namespace: default
      # All workloads in the "kube-system" Kubernetes namespace will be
      # scheduled onto Fargate:
      - namespace: kube-system
  - name: fp-dev
    selectors:
      # All workloads in the "dev" Kubernetes namespace matching the following
      # label selectors will be scheduled onto Fargate:
      - namespace: dev
        labels:
          env: dev
          checks: passed
```


aws

Удаление Fargate profile

```
eksctl delete fargateprofile --cluster cluster-name --name fp-9bfc77ad --wait
```

Важно! Процесс может занимать несколько минут, и если не поставить “--wait” то сразу после ответа API eksctl решит, что все закончилось.

aws

Часть 3 – важные штуки для eksctl

aws

IAM Roles for ServiceAccounts – why?

Позволяет назначать пермиссии на доступ к AWS сервисам на конкретные PODs:

“associate an IAM role with a Kubernetes service account. K8s Service account can then provide AWS permissions to the containers in any pod that uses that service account.”

Т.е. не нужно больше назначать роли на сами WorkerNodes (как было раньше).

“By default, all containers that are running on a worker node have all permissions assigned to the Amazon EKS worker node IAM role that is attached to the worker node.” (example)

aws

IAM Roles for ServiceAccounts - configuration

Как назначить через cluster_config.yaml:

- eksctl utils associate-iam-oidc-provider --config-file=<path>
- eksctl create iamserviceaccount --config-file=<path>

eksctl create iamserviceaccount поддерживает --include and --exclude.

С ними можно работать, как с NodeGroups (include/exclude/only-missing delete)

Давайте посмотрим, как это выглядит в cluster_config.yaml

aws

EKS кластер в существующей VPC – вы отвечаете за Networking setup

- All subnets in the same VPC, within the CIDR range
- Sufficient number of subnets (minimum 2, in different AZs)
- Internet (IGW) and/or NAT gateways are configured correctly
- Routing tables have correct entries and the network is functional
- tagging of **subnets**
  - kubernetes.io/cluster/<name> tag set to either **shared** or **owned**
  - kubernetes.io/role/internal-elb tag set to 1 for private subnets

aws

EKS кластер в существующей VPC – config.yaml

NEW: all public subnets should have the property **MapPublicIpOnLaunch** enabled (i.e. Auto-assign public IPv4 address in the AWS console)

```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-test
  region: us-east-1
vpc:
  id: "vpc-11111"
  cidr: "152.28.0.0/16"
  subnets:
    private:
      us-east-1d:
          id: "subnet-1111"
          cidr: "152.28.152.0/21"
      us-east-1c:
          id: "subnet-11112"
          cidr: "152.28.144.0/21"
      us-east-1a:
          id: "subnet-11113"
          cidr: "152.28.136.0/21"
```


aws

Часть 4 – продакшн штуки на eksctl

aws

Список важного для продакшена

1. Проверить версии системных плагинов (CoreDNS, VPC CNI, etc)
    - Обновить, если требуется
2. Настроить kubelet configuration
3. Процедура обновления кластера до минорной версии
4. Security hardening продакшн кластера (e.g. private networking, restrict public access, etc.)

aws

1. Проверить плагины на предмет последних версий

CoreDNS: default=1.6.6 (upto EKS 1.16 link), latest=1.6.9 (link)

Проверить:

```
kubectl describe deployment coredns --namespace kube-system | grep Image | cut -d "/" -f 3
```

Обновить: link

VPC CNI: latest=1.6.3 (link)

Проверить:

```
kubectl describe daemonset aws-node -n kube-system | grep Image | cut -d "/" -f 2
```

Обновить: link

aws

2. Kubelet конфигурация

Почему важно это настроить:

"In the case of resource starvation the kubelet might not be able to evict pods and eventually make the node become ***NotReady***" – это значит, что мы можем неожиданно под нагрузкой «потерять» ноды из кластера – они будут заменены на новые.

Link to all params

systemReserved: resources reserved for kubernetes system components.

Currently *cpu*, *memory* and *local storage for root file system* are supported.

kubeReserved: reserved for the host level system threads and kubernetes related threads.

```
nodeGroups:
  - name: ng-1
    instanceType: m5a.xlarge
    desiredCapacity: 1
    kubeletExtraConfig:
        kubeReserved:
            cpu: "300m"
            memory: "300Mi"
            ephemeral-storage: "1Gi"
        kubeReservedCgroup: "/kube-reserved"
        systemReserved:
            cpu: "300m"
            memory: "300Mi"
            ephemeral-storage: "1Gi"
        evictionHard:
            memory.available:  "200Mi"
            nodefs.available: "10%"
        featureGates:
            DynamicKubeletConfig: true
            RotateKubeletServerCertificate: true
```

3. EKS cluster upgrade #1: обновление Control Plane

Меняете версию в cluster_config.yaml и запускаете upgrade:

```
eksctl upgrade cluster --config-file cluster_config.yaml
```


aws

3. EKS cluster upgrade #2: обновление системных PODs

Kube-proxy:

```
eksctl utils update-kube-proxy
```

AWS-node (VPC CNI):

```
eksctl utils update-aws-node
```

CoreDNS:

```
eksctl utils update-coredns
```

(на ту, которую поддерживает EKS, для 1.16=1.6.6)

Проверить, что все запустилось и работает:

```
kubectl get pods -n kube-system
```


aws

3. EKS cluster upgrade #3: обновление NodeGroups

Помним, что NodeGroups – immutable, для обновления базового AMI нужно:

В файле конфигурации поменять только имя NodeGroups (добавить “-v2”)

- Создать новые NodeGroups для тех, которые будем обновлять
    - `eksctl create nodegroup --config-file=<path>`
- Удалить старые после перемещения подов
    - `eksctl delete nodegroup --config-file=<path> --only-missing`

Важно! This will drain all pods from that nodegroup before the instances are deleted. All nodes are cordoned and all pods are evicted from a nodegroup on deletion. –> Поэтому не страшно! ☺

Для Managed NodeGroups все проще:

```
eksctl upgrade nodegroup --name={ng-name} --cluster={name}
```


aws

4. EKS cluster NodeGroups - закрываем от внешнего мира

По-умолчанию (если ничего не настроить) eksctl создает кластер в новой VPC: CIDR: 192.168.0.0/16, 8x (/19) subnets (3 private, 3 public & 2 reserved).

”Initial nodegroup is created in public subnets, with SSH access disabled unless --allow-ssh is specified.

However, this implies that each of the EC2 instances in the initial nodegroup gets a public IP and can be accessed on ports 1025 - 65535, which is not insecure in principle, but some compromised workload could risk an access violation.”

Выдержка из eksctl документации

Hardening: используем privateNetworking: true (yaml)

aws

4. EKS hardening: API endpoints & access CIDRs

Restric Public (internet) access to Kubectl API:

```
eksctl utils update-cluster-endpoints -f cluster_config.yaml --approve
```

Restrict access to Kubectl API from internet to limited CIDRs:

```
eksctl utils set-public-access-cidrs -f eksctl-private.yaml
```


aws

Часть 5 – остальное и новое на eksctl

aws

Разное, но полезное #1

Custom AMIs

- Ubuntu1804, Amilinux2, ***handmade***

Enable CloudWatch logs for Control Plane: “*”, “all”, или конкретные types:

```
cloudWatch:
  clusterLogging:
    enableTypes:
      - "audit"
      - "authenticator"
```

Разные примеры eksctl yaml:
https://github.com/weaveworks/eksctl/tree/master/examples

aws

Разное, но полезное #2

- Override instanceName in NodeGroups
- Manage SSH keys (from local, from EC2, inline)
- Enable CSI drivers support (EFS, EBS, FsX)
- Volume encryption in NodeGroups instances
- K8s secrets encryption with KMS keyARN
- NodeGroups: custom clusterDNS (one) or kubeletExtraConfig: clusterDNS (multiple)

Gitops (via Flux Helm Operator)

aws

Новое! Поддержка Bottlerocket OS

```
nodeGroups:
  - name: ng1-public
    instanceType: m5.xlarge
    desiredCapacity: 4
    amiFamily: Bottlerocket
    ami: auto-ssm
    labels:
      "network-locality.example.com/public": "true"
    bottlerocket:
      enableAdminContainer: true
      settings:
        motd: "Hello, eksctl!"
```

Supported regions (AMI):

- ap-northeast-1
- ap-south-1
- eu-central-1 (Fra)
- us-east-1
- us-west-2

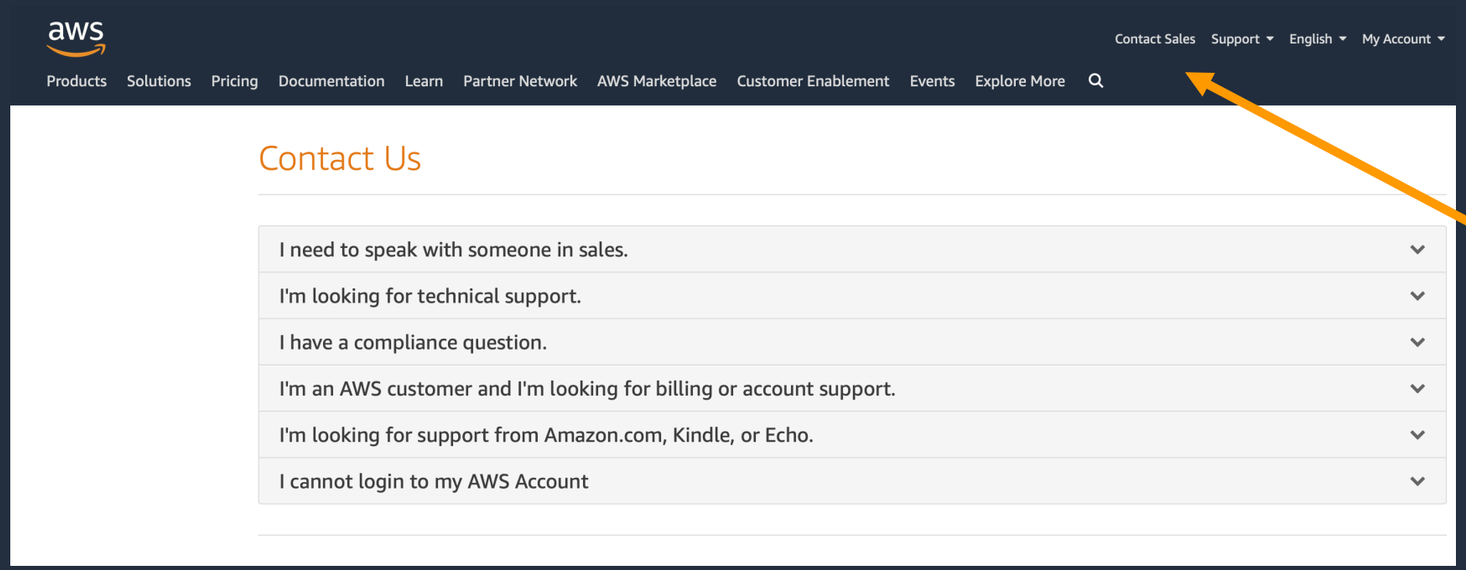
https://github.com/weaveworks/eksctl/blob/master/examples/20-bottlerocket.yaml

aws

Q&A

Александр Изюмов, AWS Sr Solutions Architect
aiziumov@amazon.de

Мы перезапустили AWS RUS блог: https://aws.amazon.com/ru/blogs/rus/



Contact Sales == Contact us!

https://aws.amazon.com/