# ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

# Методи та засоби автоматичного масштабування Kubernetes-кластеру в хмарному середовищі AWS

Кваліфікаційна робота
Другий (магістерський) рівень

Помелуйко Д.А.,
студ. гр. СПм-23-3

Керівник:
Єрьоміна Н.С.,
ст. викл.

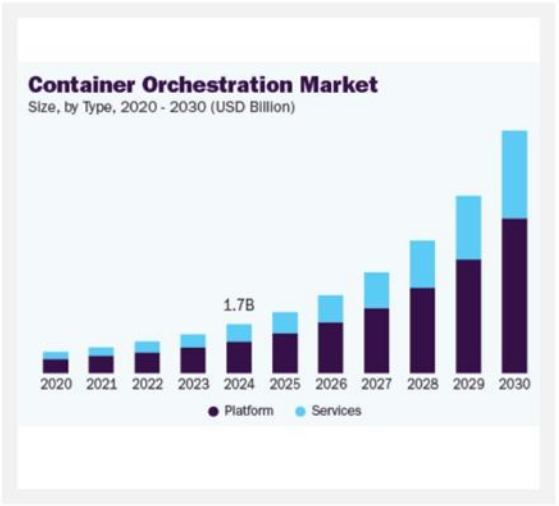
## МЕТА І ЗАДАЧІ РОБОТИ

**Мета**

Дослідження та впровадження методів автоматичного масштабування **Kubernetes**-кластеру в хмарному середовищі **Amazon Web Services** (AWS) із використанням сучасних засобів масштабування.

**Задачі**

- аналіз проблеми та предметної області;
- огляд засобів масштабування;
- розробка методів масштабування;
- реалізація та оцінка запропонованих методів.




* графік зростання капіталізації ринку оркестрації контейнерів, джерело – GrandViewResearch

## АКТУАЛЬНІСТЬ ПРОБЛЕМИ

- стрімкий розвиток та поширення мікросервісної архітектури;
- широке впровадження контейнеризації застосунків та потреба у платформах оркестрації контейнерів;
- перенесення робочих навантажень до хмарних середовищ;
- потреба у гнучкому, ефективному та автоматичному масштабуванню робочого навантаження.

## ВИКОРИСТОВУВАНІ ТЕХНОЛОГІЇ

HELM

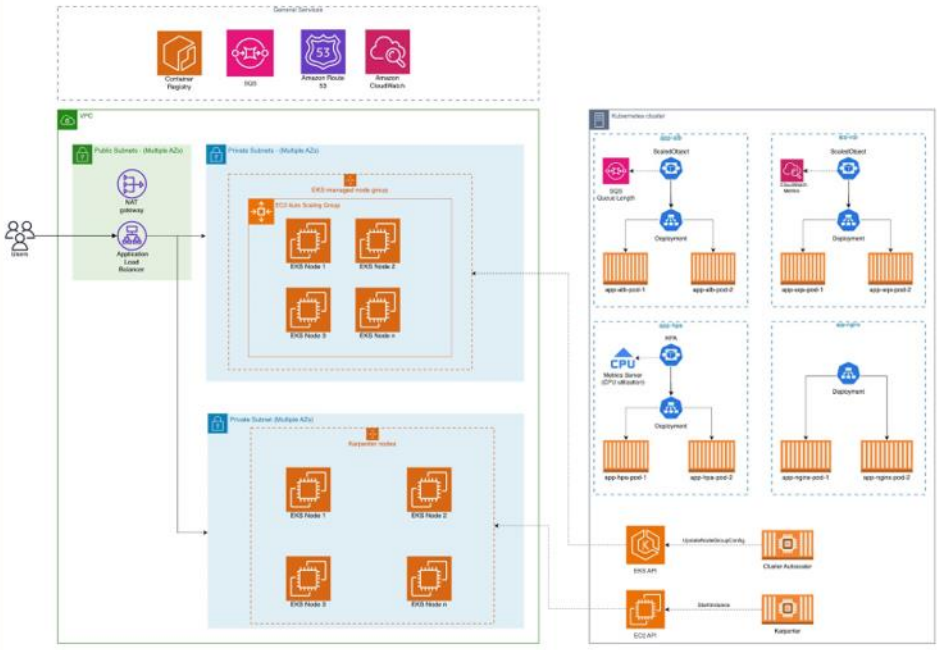
Prometheus

Grafana

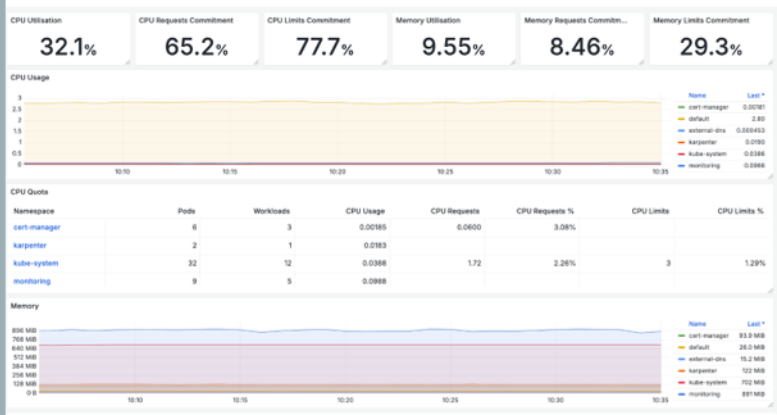
HashiCorp Terraform

aws + Kubernetes

HLD
ІНФРАСТРУКТУРИ
СЕРЕДОВИЩА
AWS



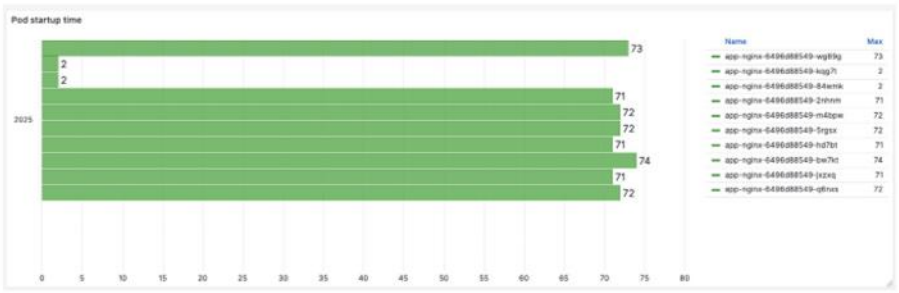
PROMETHEUS
&
GRAFANA



Експериментальне дослідження

Сценарій 1

Cluster Autoscaler та Karpenter

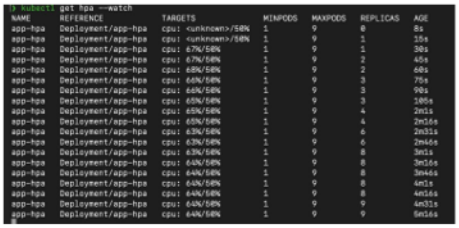




Експериментальне дослідження

Сценарій 2

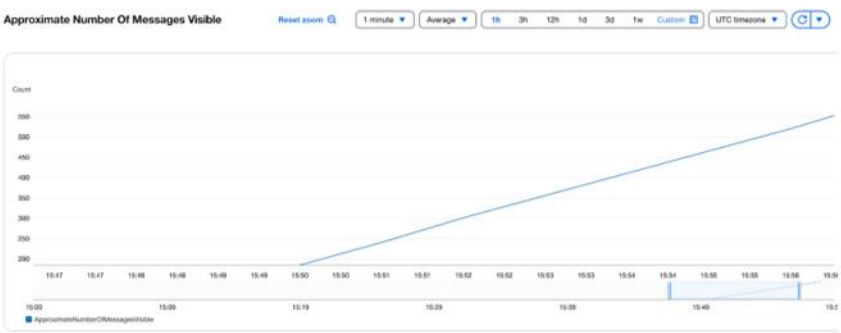
Масштабування за навантаженням ЦП

Експериментальне дослідження

Сценарій 3

Масштабування по довжині SQS черги




```
❯ python producer.py
INFO:botocore.credentials:Found credentials in shared credentials file: ~/.aws/credentials
INFO:sqs-producer:Queue length before sending: 0
INFO:sqs-producer:Sending 5 messages...
INFO:sqs-producer:Sent message ID: d7f857a0-ba26-45fe-abf4-938ff8d2fa75 | Payload: {'event_id': 'event-1', 'processing_time': 4}
INFO:sqs-producer:Sent message ID: 4bcbc706-3fdd-4e2b-8e68-474b4ab289d0 | Payload: {'event_id': 'event-2', 'processing_time': 2}
INFO:sqs-producer:Sent message ID: ff4cff1e-d75b-4c99-b218-0981bade1190 | Payload: {'event_id': 'event-3', 'processing_time': 1}
INFO:sqs-producer:Sent message ID: a16ae487-9811-4126-bb15-1a50eddd5032 | Payload: {'event_id': 'event-4', 'processing_time': 2}
INFO:sqs-producer:Sent message ID: 65fe75db-fcfe-4f9c-9a63-08c6e3e0a0f9 | Payload: {'event_id': 'event-5', 'processing_time': 5}
INFO:sqs-producer:Queue length before sending: 5
INFO:sqs-producer:Sending 5 messages...
INFO:sqs-producer:Sent message ID: 98393eb9-f030-41ef-a4fd-17cffdf48d07 | Payload: {'event_id': 'event-6', 'processing_time': 5}
INFO:sqs-producer:Sent message ID: 9e0bab93-1885-4d49-942e-d6221fd48033 | Payload: {'event_id': 'event-7', 'processing_time': 2}
INFO:sqs-producer:Sent message ID: bcaff9bd-8bc9-48dc-9a5e-bfd36a0229ed | Payload: {'event_id': 'event-8', 'processing_time': 4}
INFO:sqs-producer:Sent message ID: 4149798d-3f88-4b6f-95d4-7232986ef15c | Payload: {'event_id': 'event-9', 'processing_time': 3}
INFO:sqs-producer:Sent message ID: dc50a441-f331-4ab9-bd1a-bf9177e137cd | Payload: {'event_id': 'event-10', 'processing_time': 3}
INFO:sqs-producer:Queue length before sending: 10
INFO:sqs-producer:Sending 5 messages...
INFO:sqs-producer:Sent message ID: 8c293f1b-a558-4a45-8bf1-85695c07fab2 | Payload: {'event_id': 'event-11', 'processing_time': 4}
INFO:sqs-producer:Sent message ID: f05958a9-9d58-46ef-8d6a-80895dd25caa | Payload: {'event_id': 'event-12', 'processing_time': 0}
INFO:sqs-producer:Sent message ID: e37c05ad-a14c-4c20-9cd9-4b8832c7b2e9 | Payload: {'event_id': 'event-13', 'processing_time': 0}
INFO:sqs-producer:Sent message ID: 596f941c-ef71-42d7-aced-c45256aa81fc | Payload: {'event_id': 'event-14', 'processing_time': 2}
INFO:sqs-producer:Sent message ID: cf36c3e6-c3a8-4e2d-a244-e287ede7bf17 | Payload: {'event_id': 'event-15', 'processing_time': 4}
INFO:sqs-producer:Queue length before sending: 15
INFO:sqs-producer:Sending 5 messages...
```

Експериментальне дослідження

Сценарій 3

Масштабування по довжині SQS черги

Експериментальне дослідження

Сценарій 4

Масштабування по кількості запитів ALB


## ВИСНОВКИ

Розглянуто сучасні методи та інструменти автоматичного масштабування кластерів Kubernetes у хмарному середовищі AWS.

Розроблено методи масштабування кластеру, які є ефективнішими за традиційні, та дозволяють масштабувати кластер використовуючи зовнішні метрики та події.

Проведено експериментальні дослідження над реалізованими методами, які демонструють їх високий рівень автоматизації, адаптивності масштабованості та ефективності.

Результати досліджень опубліковано у журналі «Системи управління, навігації та зв'язку» Помелуйко Д. А., Єрьоміна Н. С. та ін.. Методи масштабування kubernetes кластеру в хмарному середовищі. Системи управління, навігації та зв'язку. 2025. No2, с. 176 – 180.

Отримані результати та розроблену модель масштабування можливо впроваджувати в існуючі та нові системи для побудови ефективних та адаптивних хмарних середовищ з **Kubernetes** кластерами.

# ДОДАТОК Б

## Конфігураційні файли

### Б.1 HCL код Terraform

```
module "vpc" {
 source  = "terraform-aws-modules/vpc/aws"
 version = "~> 5.19.0"


 name = local.basename
 cidr = var.vpc_primary_cidr


 azs             = local.zone_names
 public_subnets  = local.vpc_public_subnet_cidrs
 private_subnets = local.vpc_private_subnet_cidrs


 enable_nat_gateway = true
 single_nat_gateway = true


 public_subnet_tags = {
   "kubernetes.io/role/elb" = "1"
 }


 private_subnet_tags = {
   "kubernetes.io/role/internal-elb" = "1"
   "karpenter.sh/discovery"          = local.eks_cluster_name
 }
}




# EKS Module
module "eks" {
 source  = "terraform-aws-modules/eks/aws"
 version = "~> 20.36.0"


 cluster_name    = local.eks_cluster_name
 cluster_version = var.eks_cluster_version


 cluster_endpoint_public_access           = true
 enable_cluster_creator_admin_permissions = true
```

```
access_entries = local.eks_access_entries


cluster_addons = {
  coredns = {
    most_recent = true
    configuration_values = jsonencode({
      tolerations = local.eks_system_node_group_tolerations
      autoScaling = {
        enabled = true
      }
    })
  }
  kube-proxy = {
    most_recent = true
  }
  vpc-cni = {
    most_recent    = true
    before_compute = true
    configuration_values = jsonencode({
    })
  }
  eks-pod-identity-agent = {
    most_recent    = true
    before_compute = true
  }
  aws-ebs-csi-driver = {
    most_recent = true
    configuration_values = jsonencode({
      controller = {
        tolerations = local.eks_system_node_group_tolerations
      }
      node = {
        tolerations = local.eks_system_node_group_tolerations
      }
      defaultStorageClass = {
        enabled = true
      }
    })
    pod_identity_association = [{
      role_arn        = aws_iam_role.ebs_csi_driver.arn
      service_account = "ebs-csi-controller-sa"
    }]
  }
  external-dns = {
    most_recent = true
    configuration_values = jsonencode({
      tolerations   = local.eks_system_node_group_tolerations
      domainFilters = [aws_route53_zone.public.name]
      policy        = "upsert-only"
    })
```

```
    pod_identity_association = [{
      role_arn        = aws_iam_role.external_dns.arn
      service_account = "external-dns"
    }]
  }
  metrics-server = {
    most_recent = true
    configuration_values = jsonencode({
      tolerations = local.eks_system_node_group_tolerations
    })
  }
  cert-manager = {
    most_recent = true
    configuration_values = jsonencode({
      tolerations  = local.eks_system_node_group_tolerations
      nodeSelector = local.eks_system_node_selector
      cainjector = {
        tolerations  = local.eks_system_node_group_tolerations
        nodeSelector = local.eks_system_node_selector
      }
      webhook = {
        tolerations  = local.eks_system_node_group_tolerations
        nodeSelector = local.eks_system_node_selector
      }
    })
  }
}


# Network configuration
vpc_id                   = module.vpc.vpc_id
subnet_ids               = module.vpc.private_subnets
control_plane_subnet_ids = module.vpc.private_subnets


# KMS key
create_kms_key = true
kms_key_administrators = [
  local.deploy_role
]


create_cluster_security_group = true
create_node_security_group    = true


cluster_security_group_additional_rules = {
}


node_security_group_tags = merge(local.tags, {
  "karpenter.sh/discovery" = local.eks_cluster_name
})
```

```
 node_security_group_additional_rules = {
   ingress_cluster_metrics_server = {
     description                   = "Cluster API to node
metrics server"
     protocol                      = "tcp"
     from_port                     = 10251
     to_port                       = 10251
     type                          = "ingress"
     source_cluster_security_group = true
   }
 }


 eks_managed_node_group_defaults = {
   ami_type = "AL2023_x86_64_STANDARD"


   ebs_optimized     = true
   enable_monitoring = true


   block_device_mappings = {
     xvda = {
       device_name = "/dev/xvda"
       ebs = {
         volume_size           = 50
         volume_type           = "gp3"
         encrypted             = true
         kms_key_id            = module.kms_ebs.key_arn
         delete_on_termination = true
       }
     }
   }


   metadata_options = {
     http_put_response_hop_limit = 1
     http_endpoint               = "enabled"
     http_tokens                 = "required"
   }


   create_iam_role = true
   iam_role_attach_cni_policy = false
   iam_role_additional_policies = {
     AmazonEC2ContainerRegistryReadOnly =
"arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
     AmazonEKSWorkerNodePolicy          =
"arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
     AmazonSSMManagedInstanceCore       =
"arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
```

```
  }
}


eks_managed_node_groups = {
  system = {
    instance_types = ["r5.large"]


    min_size     = 1
    desired_size = 2
    max_size     = 3


    ebs_optimized     = true
    enable_monitoring = true


    labels = {
      pool = "system"
    }


    tags = {
      pool = "system"
    }


    taints = {
      system = {
        key    = "dedicated"
        value  = "system"
        effect = "NO_SCHEDULE"
      }
    }


    iam_role_description = "EKS-managed system node group role"
  }
  worker = {
    instance_types = ["m5.large"]


    min_size     = 0
    desired_size = 0
    max_size     = 10


    labels = {
      pool = "worker-ng"
    }
```

```
      tags = {
        pool = "worker-ng"
      }


      taints = {
        sandbox = {
          key    = "dedicated"
          value  = "worker-ng"
          effect = "NO_SCHEDULE"
        }
      }


      iam_role_description = "EKS-managed worker node group role"
    }
  }


  depends_on = [module.vpc]
}


module "karpenter" {
  source  = "terraform-aws-modules/eks/aws//modules/karpenter"
  version = "20.36.0"


  cluster_name    = module.eks.cluster_name
  service_account = "karpenter"


  namespace = "karpenter"


  enable_pod_identity             = true
  create_pod_identity_association = true


  create_iam_role       = true
  iam_role_name         = "${local.basename}-karpenter-role"
  enable_v1_permissions = true


  create_node_iam_role = true
  node_iam_role_attach_cni_policy = false
  node_iam_role_name              = "${local.basename}-karpenter-
node-role"
  node_iam_role_additional_policies = {
    AmazonSSMManagedInstanceCore =
"arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
  }
```

```
 tags = local.tags
}


resource "helm_release" "aws_lb_controller" {
 name       = "aws-load-balancer-controller"
 repository = "https://aws.github.io/eks-charts"
 chart      = "aws-load-balancer-controller"
 namespace  = "kube-system"
 version    = "1.13.2"


 dependency_update = true


 values = [
   yamlencode({
     vpcId       = module.vpc.vpc_id
     clusterName = module.eks.cluster_name
     tolerations = local.eks_system_node_group_tolerations
     #enableServiceMutatorWebhook = false
     serviceAccount = {
       name = local.eks_aws_lb_controller_sa
     }
     serviceMonitor = {
       enabled = true
       # namespace = local.eks_prometheus_namespace
       interval = "15s"
     }
   })
 ]


 depends_on = [module.eks]
}


resource "helm_release" "karpenter_crd" {
 name                = "karpenter-crd"
 namespace           = module.karpenter.namespace
 create_namespace    = true
 repository          = "oci://public.ecr.aws/karpenter"
 chart               = "karpenter-crd"
 version             = "1.5.0"
 repository_username =
data.aws_ecrpublic_authorization_token.token.user_name
 repository_password =
data.aws_ecrpublic_authorization_token.token.password


 depends_on = [module.eks]
}
```

```
resource "helm_release" "karpenter" {
 name                = "karpenter"
 namespace           = module.karpenter.namespace
 create_namespace    = true
 repository          = "oci://public.ecr.aws/karpenter"
 chart               = "karpenter"
 version             = helm_release.karpenter_crd.version
 repository_username =
data.aws_ecrpublic_authorization_token.token.user_name
 repository_password =
data.aws_ecrpublic_authorization_token.token.password


 values = [
   yamlencode({
     serviceAccount = {
       name = module.karpenter.service_account
     }
     settings = {
       clusterName       = module.eks.cluster_name
       clusterEndpoint   = module.eks.cluster_endpoint
       interruptionQueue = module.karpenter.queue_name
       featureGates = {
         spotToSpotConsolidation = true
       }
     }
     tolerations = local.eks_system_node_group_tolerations
     serviceMonitor = {
       enabled = true
     }
   })
 ]


 depends_on = [helm_release.karpenter_crd]
}


resource "helm_release" "prometheus_stack" {
 name       = "kube-prometheus-stack"
 repository = "https://prometheus-community.github.io/helm-
charts"
 chart      = "kube-prometheus-stack"
 namespace  = local.eks_prometheus_namespace
 version    = "72.6.3"


 create_namespace  = true
 dependency_update = true
```

```
 values = [
   yamlencode({
     nodeSelector = local.eks_system_node_selector
     tolerations  = local.eks_system_node_group_tolerations
     prometheusOperator = {
       admissionWebhooks = {
         patch = {
           tolerations  =
local.eks_system_node_group_tolerations
           nodeSelector = local.eks_system_node_selector
         }
       }
       nodeSelector = local.eks_system_node_selector
       tolerations  = local.eks_system_node_group_tolerations
     }
     alertmanager = {
       enabled = false
     }
     grafana = {
       enabled      = true
       nodeSelector = local.eks_system_node_selector
       tolerations  = local.eks_system_node_group_tolerations
       persistence = {
         enabled = true
         size    = "10Gi"
       }
       ingress = {
         enabled = true
         annotations = {
           "kubernetes.io/ingress.class"             = "alb"
           "alb.ingress.kubernetes.io/group.name"    =
"monitoring"
           "alb.ingress.kubernetes.io/scheme"        =
"internet-facing"
           "alb.ingress.kubernetes.io/target-type"   = "ip"
           "alb.ingress.kubernetes.io/ssl-redirect"  = "443"
           "alb.ingress.kubernetes.io/listen-ports"  =
jsonencode([{ HTTP = 80 }, { HTTPS = 443 }])
           "alb.ingress.kubernetes.io/inbound-cidrs" =
local.my_ip_cidr
         }
         hosts = ["grafana.${var.r53_domain_name}"]
         path  = "/*"
       }
       dashboardProviders = {
         "dashboardproviders.yaml" = {
           apiVersion = 1
           providers = [
             {
               name            = "default"
               orgId           = 1
               folder          = ""
               type            = "file"
```

```
        disableDeletion = false
        editable        = true
        allowUiUpdates  = true
        options = {
          path = "/var/lib/grafana/dashboards/default"
        }


      }
    ]
  }
}
dashboards = {
  default = {
    aws-cni-metrics = {
      gnetId     = 16032
      datasource = "Prometheus"
    }
    karpenter = {
      gnetId     = 20398
      revision   = 2
      datasource = "prometheus"
    }
    karpenter-overview = {
      gnetId     = 22171
      revision   = 3
      datasource = "Prometheus"
    }
    karpenter-activity = {
      gnetId     = 22172
      revision   = 3
      datasource = "Prometheus"
    }
    karpenter-performance = {
      gnetId     = 22173
      revision   = 3
      datasource = "Prometheus"
    }
    eks-control-plane = {
      gnetId     = 21192
      datasource = "Prometheus"
    }
    kubernetes-cluster-autoscaler = {
      gnetId     = 22174
      revision   = 2
      datasource = "Prometheus"
    }
    kubernetes-hpa = {
      gnetId     = 22251
      revision   = 1
      datasource = "Prometheus"
    }
  }
```

```
        }
      }
      prometheus = {
        enabled = true
        prometheusSpec = {
          serviceMonitorSelectorNilUsesHelmValues = false
          podMonitorSelectorNilUsesHelmValues     = false
          nodeSelector                            =
local.eks_system_node_selector
          tolerations                             =
local.eks_system_node_group_tolerations
          storageSpec = {
            volumeClaimTemplate = {
              spec = {
                resources = {
                  requests = {
                    storage = "50Gi"
                  }
                }
              }
            }
          }
        }
        ingress = {
          enabled = true
          annotations = {
            "kubernetes.io/ingress.class"             = "alb"
            "alb.ingress.kubernetes.io/group.name"    =
"monitoring"
            "alb.ingress.kubernetes.io/scheme"        =
"internet-facing"
            "alb.ingress.kubernetes.io/target-type"   = "ip"
            "alb.ingress.kubernetes.io/inbound-cidrs" =
local.my_ip_cidr
            "alb.ingress.kubernetes.io/listen-ports"  =
jsonencode([{ HTTP = 80 }, { HTTPS = 443 }])
            "alb.ingress.kubernetes.io/ssl-redirect"  = "443"
          }
          hosts = ["prometheus.${var.r53_domain_name}"]
          path  = "/*"
        }
      }
      kube-state-metrics = {
        tolerations = local.eks_system_node_group_tolerations
      }
    })
  ]


  depends_on = [module.eks]
}
```

```
resource "helm_release" "cluster_autoscaler" {
 name       = "cluster-autoscaler"
 repository = "https://kubernetes.github.io/autoscaler"
 chart      = "cluster-autoscaler"
 namespace  = "kube-system"
 version    = "9.46.6"


 values = [
   yamlencode({
     autoDiscovery = {
       clusterName = module.eks.cluster_name
     }
     awsRegion = var.aws_region
     rbac = {
       create = true
       serviceAccount = {
         create = true
         name   = local.eks_cluster_autoscaler_sa
       }
     }
     nodeSelector = local.eks_system_node_selector
     tolerations  = local.eks_system_node_group_tolerations
     extraArgs = {
       logtostderr              = true
       stderrthreshold          = "info"
       v                        = 4
       scale-down-unneeded-time = "2m"
     }
     serviceMonitor = {
       enabled = true
     }
   })
 ]


 depends_on = [module.eks]
}


resource "helm_release" "keda" {
 name       = "keda"
 repository = "https://kedacore.github.io/charts"
 chart      = "keda"
 namespace  = "kube-system"
 version    = "2.17.1"


 values = [
   yamlencode({
     clusterName = module.eks.cluster_name
```

```
      nodeSelector = local.eks_system_node_selector
      tolerations  = local.eks_system_node_group_tolerations


      prometheus = {
        operator = {
          enabled = true
          serviceMonitor = {
            enabled = true
          }
          podMonitor = {
            enabled = true
          }
        }
      }


      serviceAccount = {
        operator = {
          name = local.eks_keda_operator_sa
        }
      }
    })
 ]


 depends_on = [module.eks]
}


# EBS CSI driver
resource "aws_iam_role" "ebs_csi_driver" {
 assume_role_policy =
data.aws_iam_policy_document.pod_assume_role_policy.json
 name               = "${module.eks.cluster_name}-ebs-csi-
driver-role"
}


resource "aws_iam_role_policy_attachment" "ebs_csi_driver" {
 role       = aws_iam_role.ebs_csi_driver.name
 policy_arn = "arn:aws:iam::aws:policy/service-
role/AmazonEBSCSIDriverPolicy"
}


# External DNS
resource "aws_iam_role" "external_dns" {
 assume_role_policy =
data.aws_iam_policy_document.pod_assume_role_policy.json
 name               = "${module.eks.cluster_name}-external-dns-
role"
}
```

```
data "aws_iam_policy_document" "external_dns" {
 version = "2012-10-17"


 statement {
   effect = "Allow"
   actions = [
     "route53:ChangeResourceRecordSets"
   ]
   resources = [
     aws_route53_zone.public.arn
   ]
 }


 statement {
   effect = "Allow"
   actions = [
     "route53:ListHostedZones",
     "route53:ListResourceRecordSets"
   ]
   resources = ["*"]
 }
}


resource "aws_iam_policy" "external_dns" {
 name        = "${module.eks.cluster_name}-external-dns-policy"
 description = "This policy grants permissions to use manage
Route53 hosted zones for external dns."
 policy      = data.aws_iam_policy_document.external_dns.json
}


resource "aws_iam_role_policy_attachment" "external_dns" {
 role       = aws_iam_role.external_dns.name
 policy_arn = aws_iam_policy.external_dns.arn
}


# AWS Load Balancer controller
module "aws_lb_controller_pod_identity" {
 source  = "terraform-aws-modules/eks-pod-identity/aws"
 version = "~> 1.11.0"


 name = "${module.eks.cluster_name}-aws-lb-controller"


 attach_aws_lb_controller_policy = true
```

```
 associations = {
   (module.eks.cluster_name) = {
     cluster_name    = module.eks.cluster_name
     namespace       = "kube-system"
     service_account = local.eks_aws_lb_controller_sa
   }
 }


 tags = local.tags
}


# AWS Cluster Autoscaler
module "cluster_autoscaler_pod_identity" {
 source  = "terraform-aws-modules/eks-pod-identity/aws"
 version = "~> 1.11.0"


 name = "${module.eks.cluster_name}-cluster-autoscaler"


 attach_cluster_autoscaler_policy = true
 cluster_autoscaler_cluster_names = [module.eks.cluster_name]


 associations = {
   (module.eks.cluster_name) = {
     cluster_name    = module.eks.cluster_name
     namespace       = "kube-system"
     service_account = local.eks_cluster_autoscaler_sa
   }
 }


 tags = local.tags
}


# AWS VPC CNI
module "aws_vpc_cni_pod_identity" {
 source  = "terraform-aws-modules/eks-pod-identity/aws"
 version = "~> 1.11.0"


 name = "${module.eks.cluster_name}-vpc-cni"


 attach_aws_vpc_cni_policy = true
 aws_vpc_cni_enable_ipv4   = true
```

```
 associations = {
   (module.eks.cluster_name) = {
     cluster_name    = module.eks.cluster_name
     namespace       = "kube-system"
     service_account = "aws-node"
   }
 }


 tags = local.tags
}


# Keda Operator
module "keda_operator_pod_identity" {
 source  = "terraform-aws-modules/eks-pod-identity/aws"
 version = "~> 1.11.0"


 name = "${module.eks.cluster_name}-keda-operator"


 associations = {
   (module.eks.cluster_name) = {
     cluster_name    = module.eks.cluster_name
     namespace       = "kube-system"
     service_account = local.eks_keda_operator_sa
   }
 }


 attach_custom_policy = true
 policy_statements = [
   {
   actions = ["sqs:*"]
   resources = [
     aws_sqs_queue.app_sqs.arn
   ]
 },
 {
   actions = ["cloudwatch:*"]
   resources = ["*"]
 }
 ]


 tags = local.tags
}


# SQS application
module "app_sqs_pod_identity" {
 source  = "terraform-aws-modules/eks-pod-identity/aws"
```

```
 version = "~> 1.11.0"


 name = "${module.eks.cluster_name}-app-sqs"


 associations = {
   (module.eks.cluster_name) = {
     cluster_name    = module.eks.cluster_name
     namespace       = "default"
     service_account = "app-sqs"
   }
 }


 attach_custom_policy = true
 policy_statements = [{
   actions = ["sqs:*"]
   resources = [
     aws_sqs_queue.app_sqs.arn
   ]
 }]


 tags = local.tags
}


resource "aws_ecr_repository" "app_sqs" {
 name = "${local.basename}-app-sqs"
}


locals {
 basename    = "${var.project_name}-${var.project_env}"
 deploy_role = data.aws_iam_session_context.current.issuer_arn
 aws_account = data.aws_caller_identity.current.account_id


 tags = {
   Project     = var.project_name
   Environment = var.project_env
   Managed_by  = "terraform"
 }


 ### VPC
 zone_names               =
slice(data.aws_availability_zones.available.names, 0,
var.total_azs)
 vpc_subnet_cidrs         = cidrsubnets(var.vpc_primary_cidr, 5,
5, 5, 2, 2, 2)
 vpc_public_subnet_cidrs  = slice(local.vpc_subnet_cidrs, 0,
```

```
var.total_azs)
 vpc_private_subnet_cidrs = slice(local.vpc_subnet_cidrs,
var.total_azs, var.total_azs * 2)


 ### EKS
 eks_cluster_name = local.basename
 eks_system_node_group_tolerations = [
   {
     key      = "dedicated"
     value    = "system"
     operator = "Equal"
   }
 ]
 eks_system_node_selector = {
   pool = "system"
 }
 eks_prometheus_namespace = "monitoring"
 eks_access_entries = merge(
   {
     for k, v in var.eks_access_entries : k => {
       principal_arn =
data.aws_iam_role.eks_access_entries[k].arn


       policy_associations = {
         for kk, vv in v.policies : kk => {
           policy_arn = "arn:aws:eks::aws:cluster-access-
policy/${vv.name}"
           access_scope = {
             type       = vv.scope
             namespaces = vv.namespaces
           }
         }
       }
     }
 })
 eks_aws_lb_controller_sa  = "aws-lb-controller-sa"
 eks_cluster_autoscaler_sa = "cluster-autoscaler"
 eks_keda_operator_sa      = "keda-operator"


terraform {

 required_providers {
   aws = {
     source  = "hashicorp/aws"
     version = ">=5.94.1"
   }
   kubernetes = {
     source  = "hashicorp/kubernetes"
     version = ">= 2.36.0"
   }
```

```
    helm = {
      source  = "hashicorp/helm"
      version = ">= 2.17.0"
    }
    time = {
      source  = "hashicorp/time"
      version = ">= 0.12.1"
    }
  }


  backend "s3" {
    region         = "eu-west-1"
    bucket         = "tfstate"
    key            = "terraform.tfstate"
    use_lockfile   = true
  }
}


provider "aws" {
  region = var.aws_region


  allowed_account_ids = ["12345678910"]


  assume_role {
    role_arn = var.deploy_role
  }


  default_tags {
    tags = local.tags
  }
}


provider "kubernetes" {
  host                   = module.eks.cluster_endpoint
  cluster_ca_certificate =
base64decode(module.eks.cluster_certificate_authority_data)


  exec {
    api_version = "client.authentication.k8s.io/v1beta1"
    command     = "aws"
    args        = ["eks", "get-token", "--cluster-name",
module.eks.cluster_name]
  }
}
```

```
provider "helm" {
 kubernetes {
   host                   = module.eks.cluster_endpoint
   cluster_ca_certificate =
base64decode(module.eks.cluster_certificate_authority_data)


   exec {
     api_version = "client.authentication.k8s.io/v1beta1"
     command     = "aws"
     args        = ["eks", "get-token", "--cluster-name",
module.eks.cluster_name]
   }
 }
}




### General
variable "aws_region" {
 description = "AWS region name."
 type        = string
}
variable "project_name" {
 description = "Name of project."
 type        = string
}
variable "project_env" {
 description = "Name of environment."
 type        = string
}
variable "deploy_role" {
 description = "IAM role arn for AWS providers."
}


### VPC
variable "vpc_primary_cidr" {
 description = "CIDR block for the VPC."
 type        = string
}
variable "total_azs" {
 description = "Number of AZ for subnets"
 type        = number
 default     = 3
}


### EKS
variable "eks_cluster_version" {
 type        = string
 description = "Kubernetes `<major>.<minor>` version to use for
```

```
the EKS cluster (i.e.: `1.31`)"
}


### General
aws_region   = "eu-west-1"
project_name = "eks"
project_env  = "dev"
deploy_role  = "arn:aws:iam::123455678910:role/terraform"


### VPC
total_azs        = 3
vpc_primary_cidr = "10.50.0.0/16"


### EKS
eks_cluster_version = "1.33"
```

## Б.2 Вихідний код застосунку app-sqs (consumer)

```
import time
import json
import os
import logging
import boto3
from botocore.exceptions import ClientError

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger("sqs-consumer")

QUEUE_URL = os.getenv("SQS_QUEUE_URL")
MAX_MESSAGES = int(os.getenv("MAX_MESSAGES"))

sqs = boto3.client("sqs")

def process_message(body):
    try:
        payload = json.loads(body)
        processing_time = payload.get("processing_time", 1)
        event_id = payload.get("event_id", "unknown")

        logger.info(f"Processing event {event_id} for
{processing_time} seconds...")
        time.sleep(processing_time)
        logger.info(f"Finished processing event {event_id}")
    except Exception as e:
        logger.error(f"Failed to process message: {e}")

def consume():
    while True:
        try:
            response = sqs.receive_message(
```

```
                QueueUrl=QUEUE_URL,
                MaxNumberOfMessages=MAX_MESSAGES,
                WaitTimeSeconds=10
            )

            messages = response.get("Messages", [])
            if not messages:
                logger.info("No messages in queue.")
                continue

            for message in messages:
                receipt_handle = message["ReceiptHandle"]
                body = message["Body"]

                process_message(body)

                sqs.delete_message(QueueUrl=QUEUE_URL,
ReceiptHandle=receipt_handle)
        except ClientError as e:
            logger.error(f"AWS error: {e}")
        except KeyboardInterrupt:
            logger.info("Shutting down consumer...")
            break

if __name__ == "__main__":
    consume()

FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt /app/requirements.txt

RUN pip install -r requirements.txt

COPY consumer.py /app/consumer.py

ENV VERSION=v1.0.0

EXPOSE 8080

CMD ["python", "consumer.py"]
```

## Б.3 Вихідний код застосунку app-sqs (producer)

```
import os
import json
import time
import random
import logging
import boto3

QUEUE_URL = os.getenv("SQS_QUEUE_URL")
```

```
MESSAGES_PER_INTERVAL = 5
INTERVAL_SECONDS = 5
MESSAGE_GROUP_ID = "default"

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger("sqs-producer")

sqs = boto3.client("sqs")

def get_queue_length():
    try:
        attrs = sqs.get_queue_attributes(
            QueueUrl=QUEUE_URL,
            AttributeNames=["ApproximateNumberOfMessages"]
        )
        return
int(attrs["Attributes"]["ApproximateNumberOfMessages"])
    except Exception as e:
        logger.warning(f"Failed to get queue length: {e}")
        return -1

def send_message(event_id, processing_time):
    payload = {
        "event_id": event_id,
        "processing_time": processing_time
    }

    response = sqs.send_message(
        QueueUrl=QUEUE_URL,
        MessageBody=json.dumps(payload),
        MessageGroupId=MESSAGE_GROUP_ID
    )
    logger.info(f"Sent message ID: {response['MessageId']} |
Payload: {payload}")

def generate_messages():
    counter = 0
    while True:
        queue_length = get_queue_length()
        if queue_length >= 0:
            logger.info(f"Queue length before sending:
{queue_length}")

        logger.info(f"Sending {MESSAGES_PER_INTERVAL}
messages...")
        for _ in range(MESSAGES_PER_INTERVAL):
            counter += 1
            event_id = f"event-{counter}"
            processing_time = random.randint(0, 5)
            send_message(event_id, processing_time)

        time.sleep(INTERVAL_SECONDS)
```

```
if __name__ == "__main__":
    if not QUEUE_URL:
        raise ValueError("SQS_QUEUE_URL environment variable is
required")
    generate_messages()
```