OpenTUNA

TUNA for Everyone

陈嘉杰

计算机科学与技术系

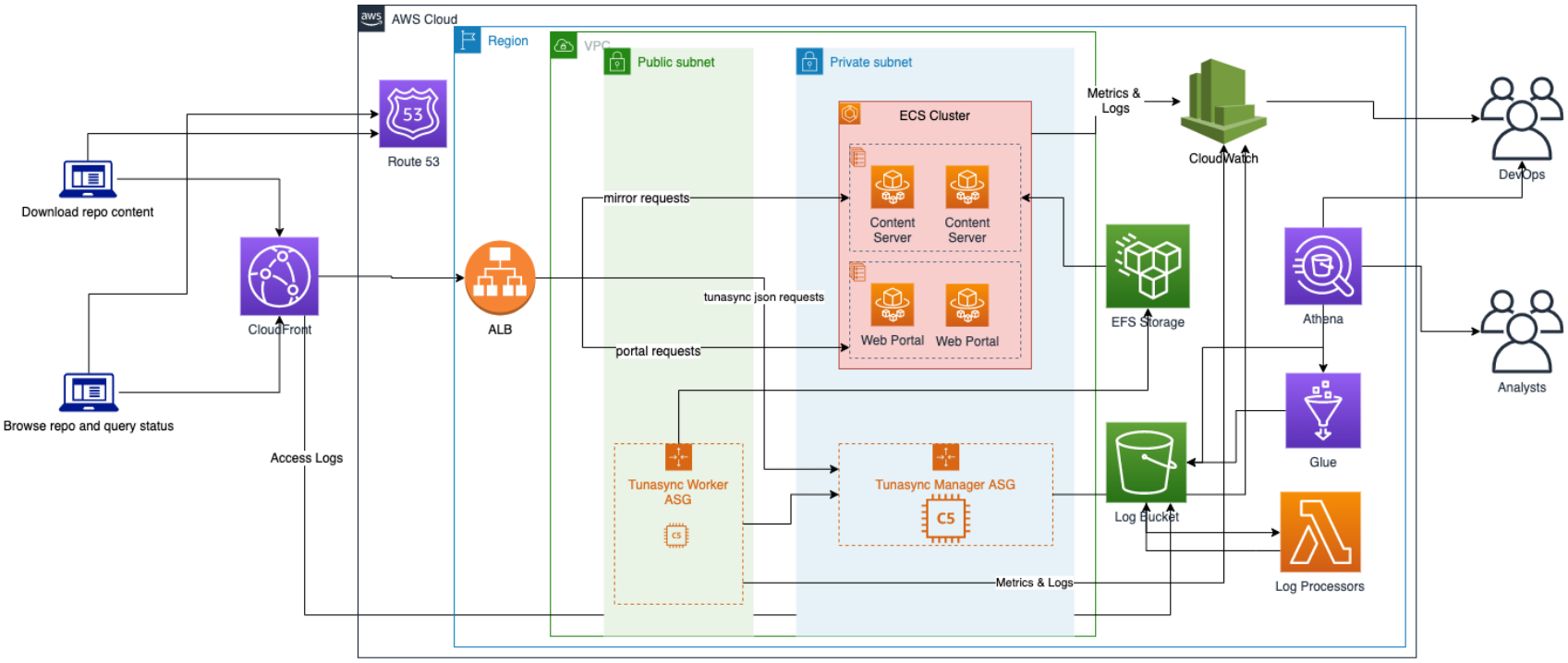
目录

Beamer 模板由 @scateu (Kang Wang) 友情提供

介绍

- OpenTUNA 照搬了 TUNA 的技术栈：mirror-web 和 tunasync。
- 针对云服务做了一些架上的调整，下面会提到。
- 运行在 AWS 中国宁夏区域。
- 代码公开在 GitHub tuna/opentuna。
- 本 Tunight 不隶属于 AWS 或其子公司。

整体架构


AWS Cloud
Region
VPC
Public subnet
Private subnet
Route 53
Download repo content
Browse repo and query status
CloudFront
ALB
mirror requests
tunasync json requests
portal requests
ECS Cluster
Content Server
Content Server
Web Portal
Web Portal
Access Logs
Tunasync Worker ASG
Tunasync Manager ASG
C5
Metrics & Logs
CloudWatch
EFS Storage
Athena
Glue
Log Bucket
Log Processors
DevOps
Analysts

AWS CDK

- 用代码描述云服务 Infra as Code
- `npx cdk deploy OpenTunaStack [args]` 就可以把代码描述的架构部署到云端
- 它的原理是生成 CloudFormation 所需要的 JSON 格式的描述，然后进行部署
- 类似 Terraform

Tunasync

- 在 EC2 中用 Auto Scaling Group 分别运行 Manager 和 Worker

```
const tunaManagerASG = new autoscaling.AutoScalingGroup(this, `${usage}ASG`, {
    instanceType: ec2.InstanceType.of(ec2.InstanceClass.C5, ec2.InstanceSize.LARGE),
    machineImage: ec2.MachineImage.latestAmazonLinux({
        generation: ec2.AmazonLinuxGeneration.AMAZON_LINUX_2,
    }),
    vpc: props.vpc,
    userData: ec2.UserData.custom(Mustache.render(userdata, newProps)),
    role: ec2Role,
    notificationsTopic: props.notifyTopic,
    minCapacity: 1,
    maxCapacity: 1,
    healthCheck: autoscaling.HealthCheck.elb({ grace: cdk.Duration.seconds(180) }),
    updateType: autoscaling.UpdateType.ROLLING_UPDATE,
    cooldown: cdk.Duration.seconds(30),
});
```

Tunasync

- 用 cloud-init 进行 EC2 内环境的配置

```
#cloud-config
repo_update: true
repo_upgrade: all
packages:
 - nfs-utils
 - amazon-efs-utils
 - amazon-cloudwatch-agent

# run commands
runcmd:
 - file_system_id_1={{&fileSystemId}}
 - efs_mount_point_1=/mnt/efs/opentuna
 - mkdir -p "${efs_mount_point_1}"
```

Tunasync

- 用 cloud-init 进行 EC2 内环境的配置
- mount EFS 存储
- 下载 tunasync 二进制
- 启动 systemd 服务
- 运行收集日志和指标的 CloudWatch Agent (基于 Telegraf)

```
"file_path": "/var/log/tunasync.log",
"log_group_name": "{{&logPrefix}}/manager",
"log_stream_name": "{instance_id}_{hostname}",
"timestamp_format": "%H: %M: %S%y%b%-d",
"timezone": "UTC"
```

Tunasync

- 代码描述了 EC2 实例的权限：SSM 和 CloudWatch
- SSM：在网页 Console 上打开一个 Shell

```
const ec2Role = new iam.Role(this, `${usage}EC2Role`, {
    assumedBy: new iam.CompositePrincipal(
        new iam.ServicePrincipal('ec2.amazonaws.com')),
    managedPolicies: [
        iam.ManagedPolicy.fromAwsManagedPolicyName('AmazonSSMManagedInstanceCore'),
        iam.ManagedPolicy.fromAwsManagedPolicyName('CloudWatchAgentServerPolicy'),
    ]
});
```

Tunasync

- 监控指标，并且发送警告到 SNS

```
const tunaWorkerAlarm = new cloudwatch.Alarm(this, 'TunaWorkerAlarm', {
    metric: runningTunaWorkerProcessMetric,
    alarmDescription: `Running Tunasync Worker Process Alarm.`,
    comparisonOperator: cloudwatch.ComparisonOperator.LESS_THAN_THRESHOLD,
    threshold: 1,
    evaluationPeriods: 3,
    treatMissingData: cloudwatch.TreatMissingData.BREACHING,
    actionsEnabled: true,
});
tunaWorkerAlarm.addAlarmAction(new cw_actions.SnsAction(props.notifyTopic));
tunaWorkerAlarm.addOkAction(new cw_actions.SnsAction(props.notifyTopic));
```

SNS 通知

- 创建了一个 SNS Topic，接收各类通知和报警
- 用 Lambda 把 SNS Topic 接受到的通知发送到 Slack
- 邮件订阅

```
const slackSubscription = new lambda.Function(this, 'slack-subscription', {
    handler: 'index.handler',
    runtime: lambda.Runtime.PYTHON_3_8,
    code: lambda.Code.fromAsset(path.join(__dirname, './lambda.d/slack-webhook')),
    environment: {
        SLACK_WEBHOOK_URL: slackHookUrl,
    },
});
this.notifyTopic.addSubscription(new sns_sub.LambdaSubscription(slackSubscription));
```

Content Server

- 用 Nginx 把 EFS 上的数据提供出来
- 采用 ECS 的 Fargate 服务，在容器里提供服务
- 本地构建 Docker Image 并上传到 ECR
- 代码描述容器的端口、映像、文件系统

Content Server

```
const container = taskDefinition.addContainer("content-server", {
    image: ecs.ContainerImage.fromDockerImageAsset(imageAsset),
    logging: new ecs.AwsLogDriver({
        streamPrefix: usage,
        // like [16/Jul/2020:02:24:46 +0000]
        datetimeFormat: "\\[%d/%b/%Y:%H:%M:%S %z\\]",
        logGroup,
    }),
});
container.addMountPoints({
    readOnly: true,
    containerPath: "/mnt/efs",
    sourceVolume: "efs-volume"
});
container.addPortMappings({
    containerPort: httpPort,
});
```

Content Server

- Auto Scale 规则：网络带宽、CPU IoWait 比例
- 容器数量变更的时候发送通知到 SNS
- ECS 集群通过 ALB 提供服务

ALB (Application Load Balancer)

- 按照路径进行 LB，分别反代到前端的 Web Portal、后端的 Tunasync Manager 和 Content Server
- 设置了特殊的规则使得 apt 可以通过 HTTP 访问，其余路径跳转到 HTTPS

```
this.managerALBTargetGroup = listener.addTargets(`${usage}TargetGroup`, {
    port: this.managerPort,
    protocol: elbv2.ApplicationProtocol.HTTP,
    targets: [tunaManagerASG],
    healthCheck: {
        path: '/ping',
    },
    slowStart: cdk.Duration.seconds(60),
    deregistrationDelay: cdk.Duration.seconds(10),
});
```

Web Portal

- 在 Docker 中用 Jekyll 构建 mirror-web
- 上传到 ECR，同样地在 Fargate 中提供服务
- 在 ALB 上设置路径

```
props.externalALBListener.addTargetGroups(`${usage}TargetGroup2`, {
    targetGroups: [webTargetGroup],
    priority: 15,
    conditions: [elbv2.ListenerCondition.pathPatterns([
        "/help/*",
        "/news/*",
        "/static/*",
        "/status/*",
    ])],
})
```

CloudFront

- 使用 CloudFront 作为 CDN，有北京、宁夏、上海、深圳的 PoP
- 日志存储在 S3 Bucket 中
- opentuna.cn 是到 CloudFront 的 ALIAS
- 用 Glue 和 Athena 进行日志分析

Glue

```
const partitionedParquetTable = new glue.CfnTable(this, "partitionedParquetTable", {
    catalogId: cdk.Aws.ACCOUNT_ID,
    databaseName: analyticsDatabase.ref,
    tableInput: {
        name: "partitioned_parquet",
        description: "Parquet format access logs as transformed from gzip version",
        tableType: "EXTERNAL_TABLE",
        parameters: { has_encrypted_data: 'false', "parquet.compression": "SNAPPY" },
        partitionKeys: partitionKeys,
        storageDescriptor: {
            outputFormat: "org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat",
            columns: storageDescCols,
            inputFormat: "org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat",
            location: cloudFrontAccessLogsBucket.s3UrlForObject(props.parquetKeyPrefix),
            serdeInfo: {
                serializationLibrary: "org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe"
            }
        }
    }
});
```

Monitor

- 除了前面提到的一些 Health Check 和 Alarm 以外，还会定时在容器中测试 index 文件的完整性
- 对各个发行版配置一个 CodeBuild
- 在 Docker 里面替换软件源，然后 apt update
- 发生错误的时候，由 Lambda 发送请求让 Tunasync Manager 重新同步

S3

- 修改了 Rubygems（约 500GB，两百万个文件）的同步工具，直接同步到 S3 bucket
- 在 CloudFront 处添加 S3 bucket 的源
- 之后可能会考虑迁移 PyPI 到 S3 存储

```
}, {
  s3OriginSource: {
    s3BucketSource: tunaRepoBucket,
    originAccessIdentity: oai,
  },
  behaviors: [{
    pathPattern: '/rubygems/gems/*',
    // 1w cache for gem specs
    defaultTtl: cdk.Duration.days(7),
  }, {
```

总结

- 把镜像站需要的各个组件部署到云的对应服务
- 针对服务的特性进行定制
- 利用 Lambda、CodeBuild、S3 等方便的工具
- 欢迎大家使用：opentuna.cn