ContaAzul

Up and Running in minutes after a DR

Como a ContaAzul age em situações de recuperação de desastres

Cássio Moreto

SRE @ ContaAzul

kassyuz

cassio.moreto@contaazul.com

ContaAzul

Joinville, SC

Plataforma para o micro e pequeno empreendedor gerenciar seu negócio.

Compra, venda, geração de NF, cobrança via Boletos, etc.

Disaster recovery plan

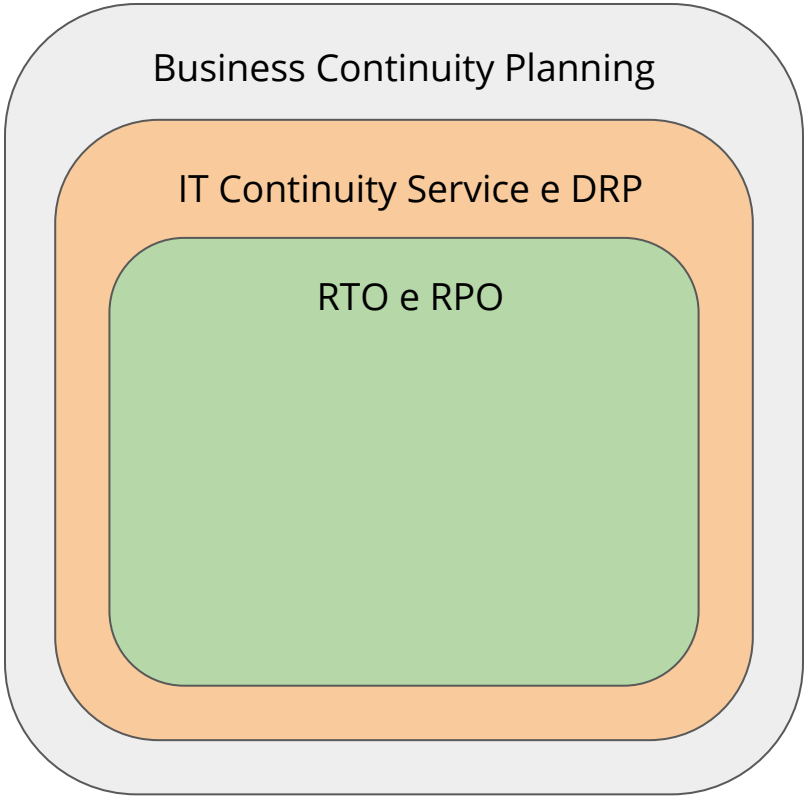
É o plano que fala como devemos agir em uma situação de desastre.

Como e por onde começar? Defina seu plano de recuperação previamente, senão...

O que planejar? Defina com seus *stackholder* o RTO e RPO

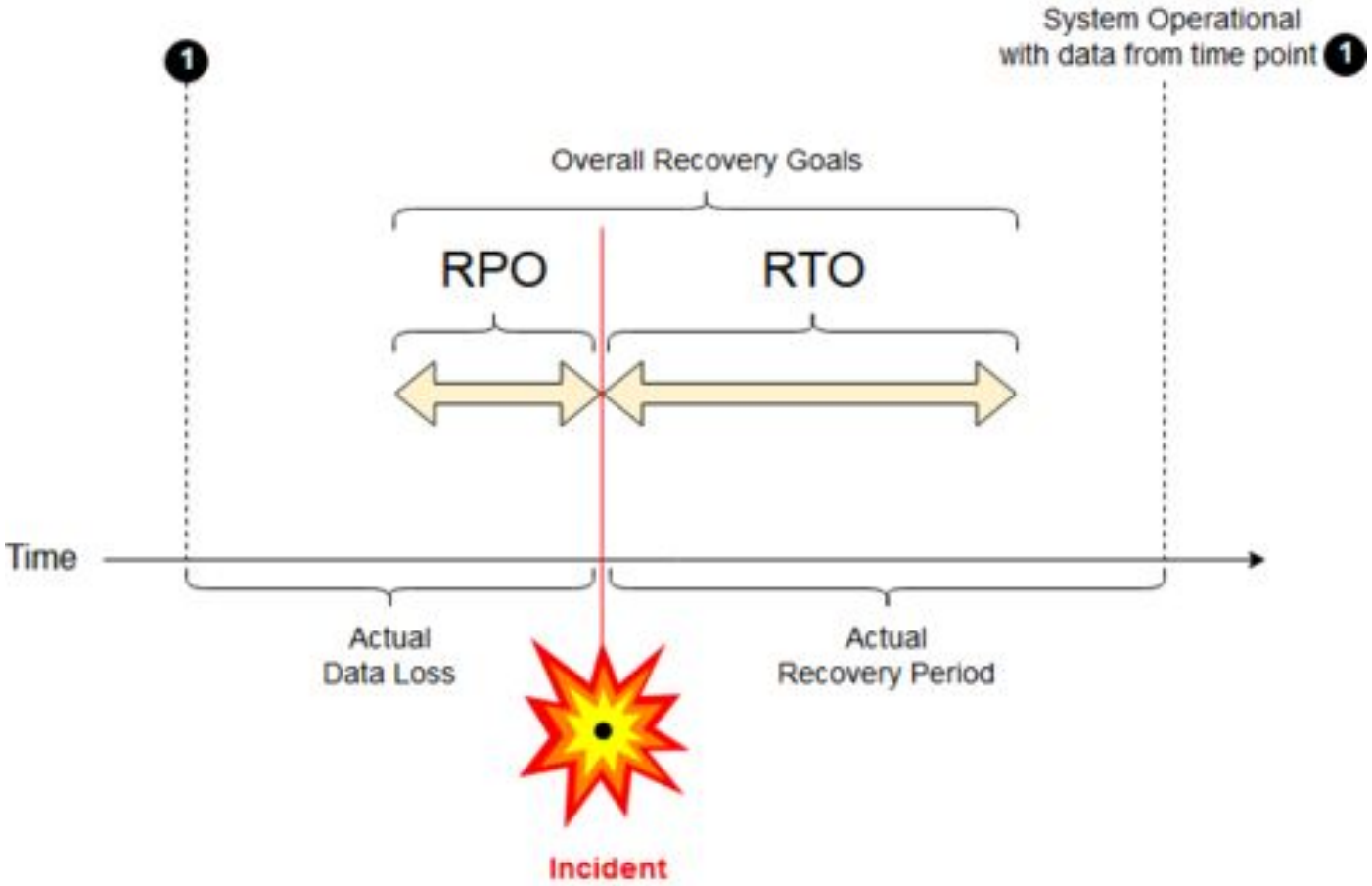
Stackholder são os interessados no software que você está desenvolvendo.

Business Continuity Planning > *IT Continuity Service* > RTO e RPO

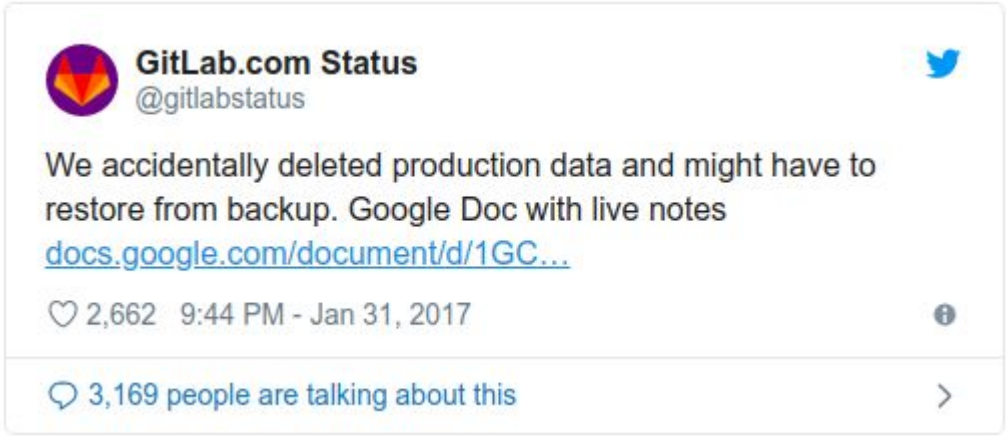


ContaAzul

RTO Recovery Time Objective
RPO Recovery Point Objective



https://en.wikipedia.org/wiki/Disaster_recovery#IT_Service_Continuity



Postmortem GitLab

https://about.gitlab.com/blog/2017/02/10/postmortem-of-database-outage-of-january-31/

O primeiro objetivo é ter tudo em código...

Mas o que?

Operações manuais que possam ser automatizadas, criação de infraestrutura, integração contínua e entrega contínua.

Transforme sua infraestrutura em código

Este é o início de tudo, assim você consegue garantir a imutabilidade, versionar seu código e ter verificações constantes dele.

Encapsular código ajuda a mitigar problemas futuros

Como fazemos na Conta Azul?

Continuous Integration

100% do nosso código de infraestrutura é testado e *deployado* **automaticamente**.

Dê autonomia para o desenvolvedor.

Porque CI é importante? Você precisa garantir que seu código esteja sempre funcionando e assim você verifica vários *Pull requests* ao mesmo tempo.

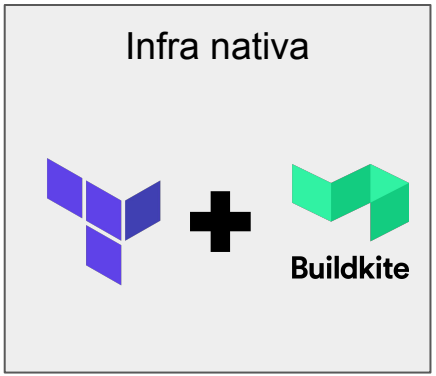
O que usamos para fazer acontecer?

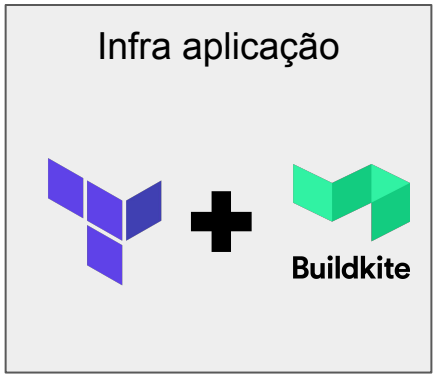
Buildkite

GitHub

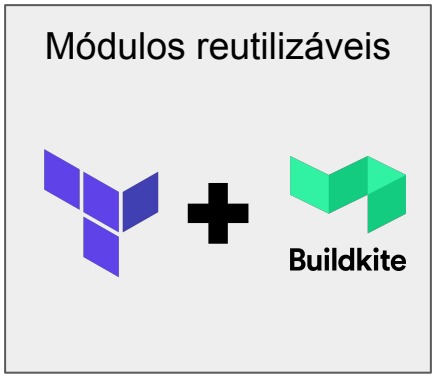
Estruturando para restaurar

Primeiro separamos infraestrutura nativa de infraestrutura de aplicação





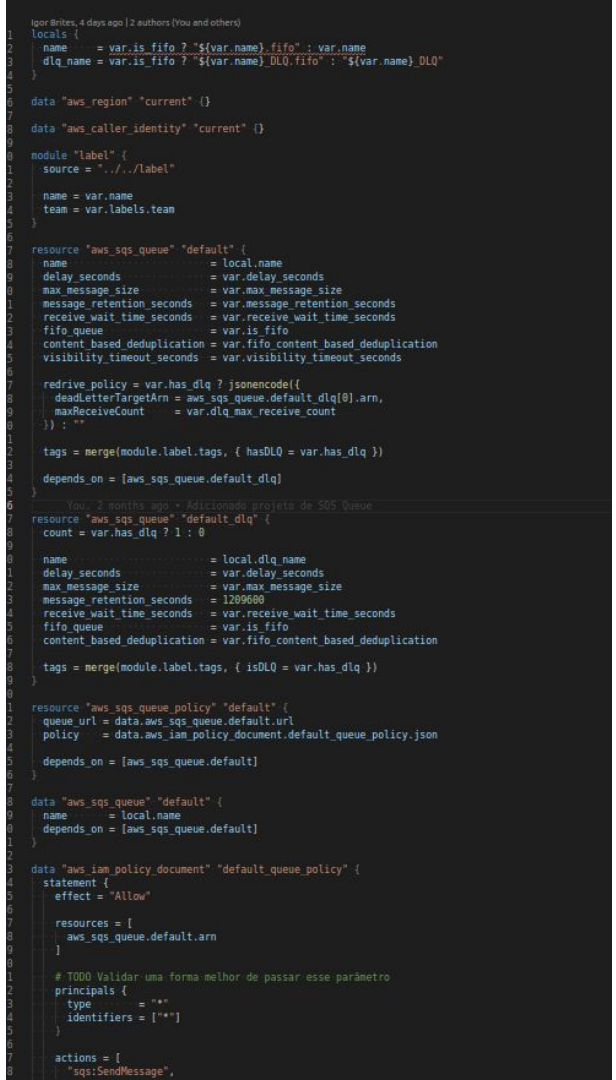
Encapsulamos o código de infra para facilitar e padronizar a criação



```hcl
module "my_queue" {
  source = "git::git@github.com:ContaAzul/terraform-modules.git//contaazul/queue?ref=v0.9.0"

  labels = {
    team = "this-crazy-team"
  }

  name = "this-crazy-queue"
  subscribe_to = [
    module.my_contaazul_topic.arn
  ]
}
```



```hcl
Igor Brites, 4 days ago | 2 authors (You and others)
locals {
  name     = var.is_fifo ? "${var.name}.fifo" : var.name
  dlq_name = var.is_fifo ? "${var.name}_DLQ.fifo" : "${var.name}_DLQ"
}

data "aws_region" "current" {}

data "aws_caller_identity" "current" {}

module "label" {
  source = "../../label"

  name = var.name
  team = var.labels.team
}

resource "aws_sqs_queue" "default" {
  name                       = local.name
  delay_seconds              = var.delay_seconds
  max_message_size           = var.max_message_size
  message_retention_seconds  = var.message_retention_seconds
  receive_wait_time_seconds  = var.receive_wait_time_seconds
  fifo_queue                 = var.is_fifo
  content_based_deduplication = var.fifo_content_based_deduplication
  visibility_timeout_seconds  = var.visibility_timeout_seconds

  redrive_policy = var.has_dlq ? jsonencode({
    deadLetterTargetArn = aws_sqs_queue.default_dlq[0].arn,
    maxReceiveCount     = var.dlq_max_receive_count
  }) : ""

  tags = merge(module.label.tags, { hasDLQ = var.has_dlq })

  depends_on = [aws_sqs_queue.default_dlq]
}
    You, 2 months ago • Adicionado projeto de SQS Queue
resource "aws_sqs_queue" "default_dlq" {
  count = var.has_dlq ? 1 : 0

  name                       = local.dlq_name
  delay_seconds              = var.delay_seconds
  max_message_size           = var.max_message_size
  message_retention_seconds  = 1209600
  receive_wait_time_seconds  = var.receive_wait_time_seconds
  fifo_queue                 = var.is_fifo
  content_based_deduplication = var.fifo_content_based_deduplication

  tags = merge(module.label.tags, { isDLQ = var.has_dlq })
}

resource "aws_sqs_queue_policy" "default" {
  queue_url = data.aws_sqs_queue.default.url
  policy    = data.aws_iam_policy_document.default_queue_policy.json

  depends_on = [aws_sqs_queue.default]
}

data "aws_sqs_queue" "default" {
  name       = local.name
  depends_on = [aws_sqs_queue.default]
}

data "aws_iam_policy_document" "default_queue_policy" {
  statement {
    effect = "Allow"

    resources = [
      aws_sqs_queue.default.arn
    ]

    # TODO Validar uma forma melhor de passar esse parâmetro
    principals {
      type        = "*"
      identifiers = ["*"]
    }

    actions = [
      "sqs:SendMessage",
```

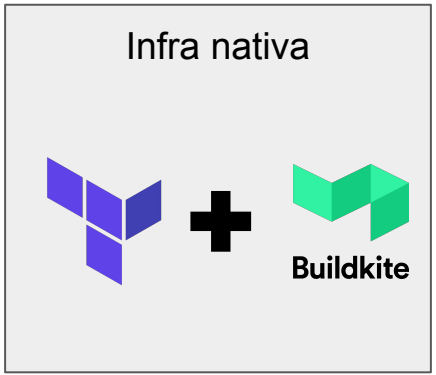

ContaAzul

O CI é executado em todos os repositórios fazendo a checagem do código.

A Restauração

Comece pelo começo, inicie recriando a infraestrutura base...

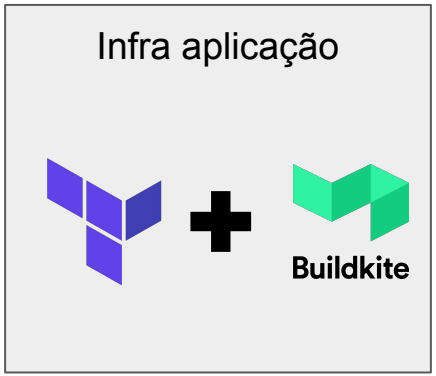
Criação das redes, Load Balancers, Bancos (ainda vazios), zonas de DNS, Cluster Kubernetes e tudo mais que você considere importante para que toda a sua aplicação comece a funcionar.



Restauração dos backups de bancos de dados

Após a base estar 100%, foque na aplicação

Crie os recursos que as aplicações irão consumir diretamente, buckets, SQS, SNS... **RDS???**



Deploy das aplicações no cluster

Dúvidas?

Cássio Moreto
SRE @ ContaAzul

github.com/kassyuz
cassio.moreto@contaazul.com
@cassiomoreto

ContaAzul