

Terraform AWS Automation Project

Полная автоматизация инфраструктуры AWS с backend'ом, locking механизмом и автоматическим управлением EC2 серверами.

Содержимое проекта

✅ Backend + Locking

- **S3 bucket** для централизованного хранения Terraform state
- **DynamoDB table** для механизма блокировки (locking)
- **Версионирование и шифрование** S3 bucket'a
- **Cross-Region Replication** между регионами

✅ EC2 Автоматизация

- **EC2 instance** с автоматическим стартом/стопом
- **Lambda functions** для управления расписанием
- **EventBridge (CloudWatch Events)** для cron-расписания
- **IAM роли и политики** для доступа к S3

✅ S3 Управление

- **Lifecycle policies** для архивирования в Glacier
- **Репликация** между регионами для отказоустойчивости
- **Автоматические бэкапы** и версионирование

Установка и запуск

Шаг 1: Подготовка

```
bash

# Клонировать репозиторий
git clone <your-repo>
cd terraform-aws-automation

# Переключиться на ветку
git checkout feature/terraform-aws-automation
```

Шаг 2: Настройка переменных

Создайте файл `terraform.tfvars`:

hcl

```
aws_region    = "eu-central-1"
replica_region = "eu-west-1"
project_name   = "terraform-automation"
environment    = "prod"

# Расписание (UTC время)
server_start_cron = "0 7 * * MON-FRI" # 9:00 Kiev time
server_stop_cron  = "0 19 * * MON-FRI" # 21:00 Kiev time

instance_type    = "t3.micro"
enable_versioning = true
enable_replication = true
```

Шаг 3: Добавить SSH ключ

Замените в `ec2.tf` строку с публичным ключом на свой:

hcl

```
resource "aws_key_pair" "ec2_key" {
  key_name   = "${var.project_name}-key"
  public_key = "ssh-rsa AAAAB3NzaC1yc2E..." # ВАШ публичный ключ
}
```

Шаг 4: Первый запуск (создание backend)

bash

```
# Закомментируйте backend блок в backend.tf для первого запуска
# Или запустите с локальным state
terraform init
terraform plan
terraform apply
```

Шаг 5: Миграция на remote backend

bash

```
# После создания S3 и DynamoDB, раскомментируйте backend.tf
# Получите данные для backend из outputs

terraform output backend_config

# Обновите backend.tf с правильными именами
# Затем мигрируйте state

terraform init -migrate-state
```

Outputs (результаты)

После `terraform apply` вы получите:

```
bash

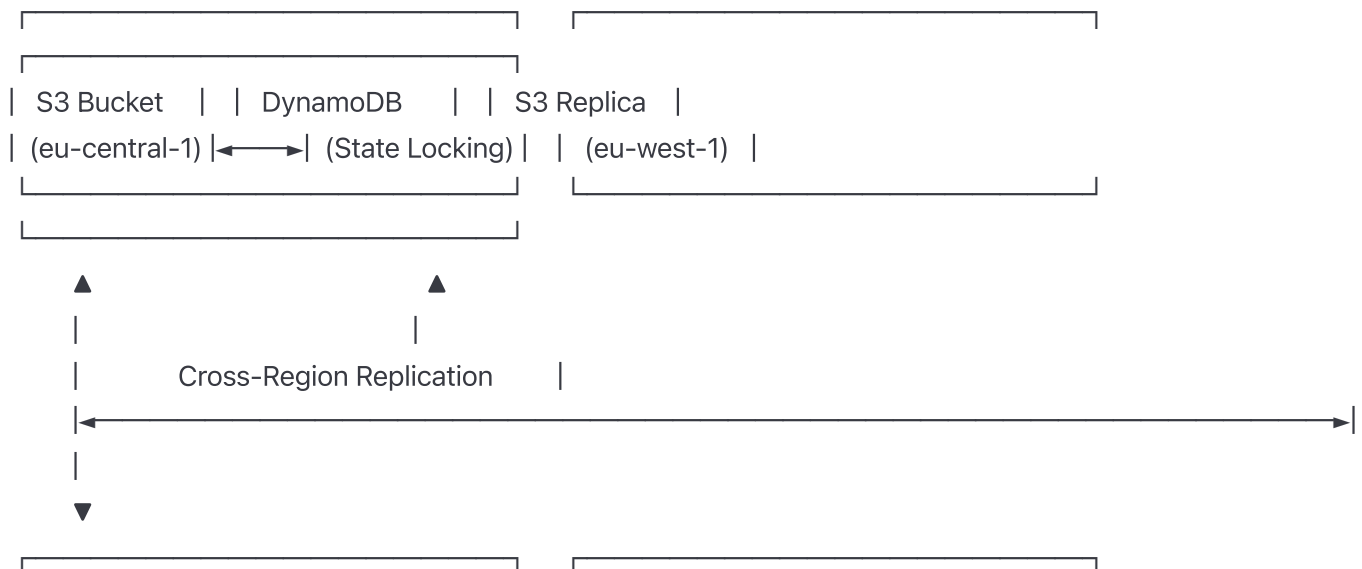
# Backend информация
s3_bucket_name = "terraform-automation-state-a1b2c3d4"
dynamodb_table_name = "terraform-automation-locks"

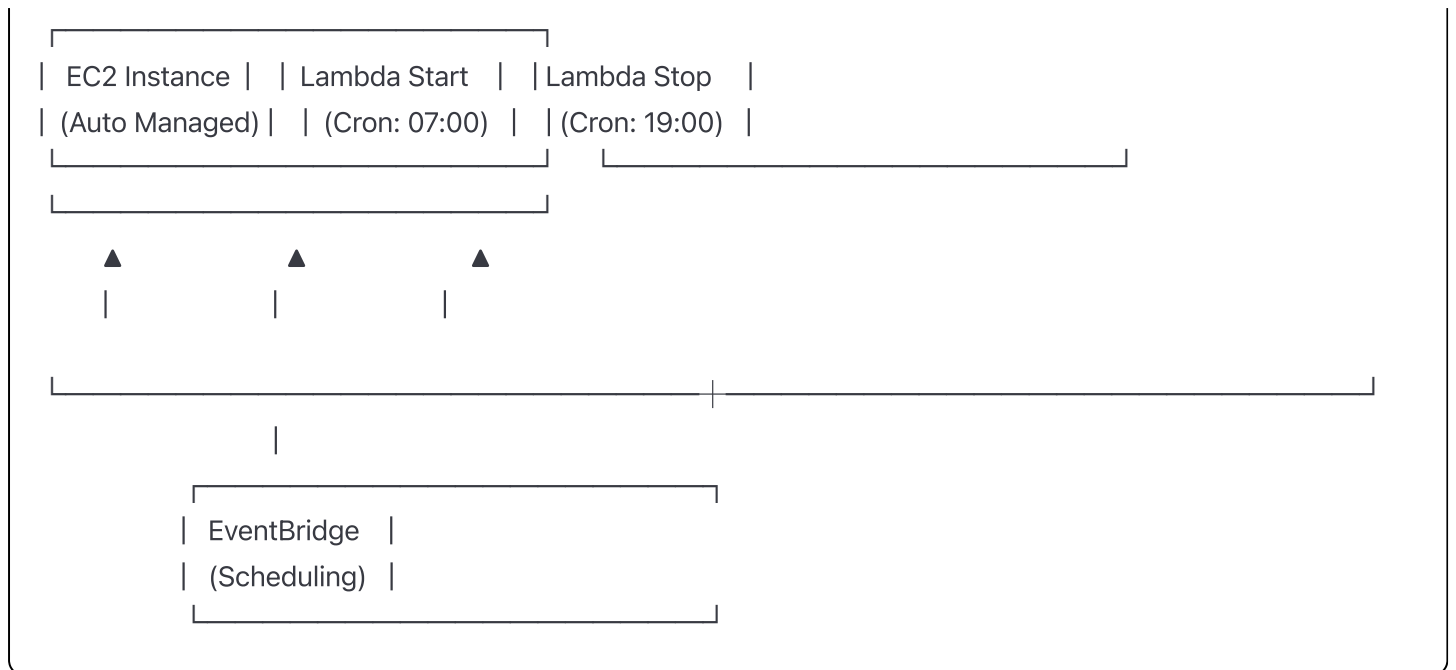
# EC2 информация
ec2_instance_id = "i-0123456789abcdef0"
ec2_public_ip = "18.185.123.45"

# Lambda функции
lambda_start_function_arn = "arn:aws:lambda:eu-central-1:..."
lambda_stop_function_arn = "arn:aws:lambda:eu-central-1:..."

# Расписание
start_schedule = "0 7 * * MON-FRI"
stop_schedule = "0 19 * * MON-FRI"
```

Архитектура





Что происходит автоматически

Ежедневное расписание:

- **07:00 UTC (09:00 Kiev)** - EC2 instance запускается
- **19:00 UTC (21:00 Kiev)** - EC2 instance останавливается
- **Только в рабочие дни** (понедельник-пятница)

Непрерывные процессы:

- **S3 replication** - объекты автоматически реплицируются
- **Lifecycle management** - старые версии архивируются в Glacier
- **State locking** - защита от конфликтов при terraform apply

Логирование:

- EC2 instance каждый час отправляет статус в S3
- Lambda функции логируют все операции start/stop
- CloudWatch содержит все логи выполнения

Безопасность

- **IAM роли** с минимальными необходимыми правами
- **S3 encryption** включено по умолчанию
- **VPC Security Groups** ограничивают доступ к EC2
- **DynamoDB encryption** включено
- **S3 public access** заблокирован



Важные моменты

1. **SSH ключ:** Обязательно замените публичный ключ в `ec2.tf`
2. **Регионы:** Убедитесь что у вас есть доступ к обоим регионам
3. **Costs:** t3.micro попадает в Free Tier, но следите за S3 и Lambda costs
4. **Security Group:** По умолчанию SSH открыт для всех IP (0.0.0.0/0)



Полезные команды

```
bash
```

```
# Проверить план изменений
```

```
terraform plan
```

```
# Применить изменения
```

```
terraform apply
```

```
# Показать все ресурсы
```

```
terraform state list
```

```
# Получить outputs
```

```
terraform output
```

```
# Уничтожить всё (осторожно!)
```

```
terraform destroy
```

```
# Проверить состояние EC2
```

```
aws ec2 describe-instances --filters "Name=tag:Name,Values=terraform-automation-auto-server"
```



Checklist для Pull Request





- ☐ SSH ключ заменён на свой
- ☐ Переменные настроены в `terraform.tfvars`
- ☐ Backend успешно сконфигурирован
- ☐ `terraform plan` выполняется без ошибок
- ☐ `terraform apply` создаёт все ресурсы
- ☐ EC2 instance может подключиться к S3
- ☐ Lambda функции работают по расписанию
- ☐ Репликация S3 настроена и работает



Готово!

После выполнения всех шагов у вас будет:

- ☒ Полностью автоматизированная инфраструктура

-  Централизованный backend с locking'ом
-  EC2 сервер с автостартом и автостопом
-  S3 с репликацией и бэкапами
-  Готовый код для Pull Request

Время выполнения: ~5-10 минут

Команда для запуска: `terraform apply` - одна команда, вся инфраструктура! 🚀