



# DevOps notes

---

**None**

*None*

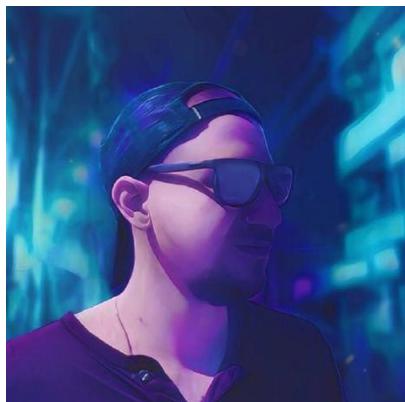
*None*

## Table of contents

---

	hey there	5
1.		
2.	01 YandexCloud	6
2.1	Утилита YC	6
2.2	Terraform	7
2.3	Security	0
3.	03 IaC	0
3.1	3.1 Yandex cloud	0
3.2	3.2 Terraform	0
3.3	3.3 Terragrunt	0
3.4	3.4 Ansible	0
4.	04 DevOps and CICD	0
4.1	4.1 Docker	0
4.2	4.2 Технология непрерывной поставки ПО	0
5.	05 Kubernetes	0
5.1	5.1 Основные компоненты k8s	0
5.2	5.2 Основные объекты кластера K8s	0
5.3	5.3 Services and Ingress	0
5.4	Ingress	0
5.5	5.4 Helm	0
5.6	5.7 Безопасность в K8s	0
5.7	5.8 KodeKloud CKAD	0
6.	06 ServiceMesh	0
6.1	6.1 Микросервисная архитектура	0
6.2	6.2 Istio	0
7.	07 Observability	0
7.1	7.1 Мониторинг	0
7.2	7.2 Логгирование	0
7.3	7.3 Трейсинг	0
8.	10 Security	0
8.1	10.1 TLS and mTLS	0
9.	11 KubernetesHardway	0
9.1	11.1 Kubernetes Hardway	0
10.	20 React	0
10.1	20.1 React	0
10.2	20.2 useState	0

11. 21 Golang .....	0
11.1 21.1 Основы Golang .....	0


[TELEGRAM](#)
[GITHUB](#)
[LINKEDIN](#)

Profile views 365

## 1. hey there

---

### About Me

I am a DevOps engineer from Saint-Petersburg

#### Completed courses:

- KodeKloud - Certified Kubernetes Application Developer (CKAD)
- Yandex practicum - DevOps
- ITMO - DevOps engineer
- Yandex practicum - Algorithm and data structure
- MIPT - Deep Learning School

#### Pet projects:

- React+Xterm - Interactive Browser Terminal playgrounds
- Python+React - Flashcard Learning Service
- Golang - telegrambot dockerization and grpc integration
- Deep learning - deploy Convolutional neural network in Heroku
- How to reach me: [Telegram](#) [Pavel](#)

#### Languages and Tools:



## 2. 01 YandexCloud

---

### 2.1 Утилита YC

Изначально создаем или изменяем профиль

Профиль — это локальная конфигурация CLI (yc), где хранятся: cloud-id, folder-id token или service-account-key.json endpoint и т.д. То есть профиль — это не сущность в Yandex Cloud, а просто набор настроек на твоём компьютере.

```
1  yc init
```

Потом можно выбрать нужный

```
1  > yc config profile list
2  default
3  hardway ACTIVE
4  terraform
5
6  > yc config profile activate terraform
7  Profile 'terraform' activated
```

Создаем сервисный аккаунт Сервисный аккаунт (Service Account, SA) — это учётка, принадлежащая не человеку, а приложению. У него: свой уникальный идентификатор (serviceAccountId), свои IAM-права (Identity and Access Management) (например storage.editor), и может быть создан статический ключ, авторизационный ключ или IAM-токен для доступа.

```
1  > yc iam service-account create --name terra
2
3  done (1s)
4  id: ajefj388645earvs656i
5  folder_id: b1gt98o9j856o777k9r5
6  created_at: "2025-11-03T14:24:19.206036538Z"
7  name: terra
```

И добавляем ему права

```
1  > yc resource-manager folder add-access-binding \
2    --service-account-name=terra \
3    --role=admin \
4    --name=default
5
6  done (2s)
7  effective_deltas:
8    - action: ADD
9      access_binding:
10        role_id: admin
11        subject:
12          id: ajefj388645earvs656i
13          type: serviceAccount
```

Добавить зеркало для провайдера Yandex vim ~/.terraformrc

```
1  provider_installation {
2    network_mirror {
3      url = "https://terraform-mirror.yandexcloud.net/"
4      include = ["registry.terraform.io/*/*"]
5    }
6    direct {
7      exclude = ["registry.terraform.io/*/*"]
8    }
9  }
```

Добавляем переменные окружения для дальнейшей работы

```
1  export YC_TOKEN=$(yc iam create-token)
2  export YC_CLOUD_ID=$(yc config get cloud-id)
3  export YC_FOLDER_ID=$(yc config get folder-id)
```

## 2.2 Terraform

### 2.2.1 Базовые понятия

Основные компоненты Terraform:

- 1 Конфигурационные файлы, или Configuration Files
- 2 Провайдеры, или Providers
- 3 Ресурсы, или Resources
- 4 Провижнёры, или Provisioners
- 5 Состояние, или State

### 2.2.2 Рабочий процесс

**1** Write или Init **2** Plan **3** Apply

Инициализация директории Код для развёртывания Managed Service for YDB:



```

1 # Создание VPC и подсети
2 resource "yandex_vpc_network" "this" {
3   name = "private"
4 }
5
6 resource "yandex_vpc_subnet" "private" {
7   name      = "private"
8   zone     = "ru-central1-a"
9   v4_cidr_blocks = ["192.168.10.0/24"]
10  network_id    = yandex_vpc_network.this.id
11 }
12
13 # Создание диска и виртуальной машины
14 resource "yandex_compute_disk" "boot_disk" {
15   name      = "boot-disk"
16   zone     = "ru-central1-a"
17   image_id = "fd8ba9d5mfvlncnt2kd" # Ubuntu 22.04 LTS
18   size     = 15
19 }
20
21 resource "yandex_compute_instance" "this" {
22   name      = "linux-vm"
23   allow_stopping_for_update = true
24   platform_id      = "standard-v3"
25   zone      = "ru-central1-a"
26
27   resources {
28     cores = "2"
29     memory = "4"
30   }
31
32   boot_disk {
33     disk_id = yandex_compute_disk.boot_disk.id
34   }
35
36   network_interface {
37     subnet_id = yandex_vpc_subnet.private.id
38   }
39 }
40
41 # Создание Yandex Managed Service for YDB
42 resource "yandex_ydb_database_serverless" "this" {
43   name      = "test-ydb-serverless"
44   location_id = "ru-central1"
45 }
46
47 # Создание сервисного аккаунта
48 resource "yandex_iam_service_account" "bucket" {
49   name = "bucket-sa"
50 }
51
52 # Назначение роли сервисному аккаунту
53 resource "yandex_resourcemanager_folder_iam_member" "storage_editor" {
54   folder_id = "b1gt98o9j856o777k9r5"
55   role      = "storage.editor"
56   member    = "serviceAccount:${yandex_iam_service_account.bucket.id}"
57 }
58
59 # Создание статического ключа доступа
60 resource "yandex_iam_service_account_static_access_key" "this" {
61   service_account_id = yandex_iam_service_account.bucket.id
62   description        = "static access key for object storage"
63 }
64
65 # Создание бакета
66 resource "yandex_storage_bucket" "this" {
67   bucket      = "test-s3-bucket-fm3oijfiwf3ore23dffoi93"
68   access_key  = yandex_iam_service_account_static_access_key.this.access_key
69   secret_key  = yandex_iam_service_account_static_access_key.this.secret_key
70
71   depends_on = [yandex_resourcemanager_folder_iam_member.storage_editor]
72 }

1 terraform {
2   required_providers {
3     yandex = {
4       source = "yandex-cloud/yandex"
5     }
6   }
7   required_version = ">= 1.00"
8 }
9
10 provider "yandex" {
11   zone      = "ru-central1-a"
12   folder_id = "b1gt98o9j856o777k9r5"
13 }

```

## Выполнение кода

```

1  terraform init
2  terraform plan
3  terraform apply
4
5  # Заменить образ
6  terraform apply -replace yandex_compute_disk.boot_disk

```

## Удалить ресурсы

```
1  terraform destroy
```

## 2.2.3 Добавляем провайдеры

```

1  terraform {
2    required_providers {
3      random = {
4        source = "hashicorp/random"
5      }
6    }
7    required_version = ">= 0.13"
8  }
9
10 resource "random_string" "bucket_name" {
11   length  = 8
12   special = false
13   upper   = false
14 }
15
16 # Создание бакета
17 resource "yandex_storage_bucket" "this" {
18   bucket      = "terraform-bucket-${random_string.bucket_name.result}"
19   access_key  = yandex_iam_service_account_static_access_key.this.access_key
20   secret_key  = yandex_iam_service_account_static_access_key.this.secret_key
21
22   depends_on = [ yandex_resourcemanager_folder_iam_member.storage_editor ]
23 }

```

## 2.2.4 Переменные в Terraform

Простые типы данных ★ String — последовательность символов Unicode, представляющая текст (например, hello). ★ Number — числовое значение, которое может представлять целые числа (например, 15) и дробные значения (например, 6,283185). ★ Bool — логическое значение (true либо false), которое может использоваться в условной логике

Сложные типы данных ★ Коллекции — для группировки схожих значений. ★ Структурные типы — для группировки потенциально несхожих значений.

Коллекции делятся на три группы: 1 Коллекция list(...) 2 Коллекция map(...) 3 Коллекция set(...)

Структурные типы данных делятся на два вида: 1 Тип данных object(...) 2 Тип данных tuple(...)

### Warning

1 ! Вы не можете назначить переменной зарезервированные имена: source, version, providers, count, for\_each, lifecycle, depends\_on, locals.



1 Аргумент `*sensitive*` Ограничивает вывод интерфейса Terraform при использовании переменной в конфигурации.

Аргумент `nullable` Контролирует, может ли переменной быть присвоено значение null.

```

1 variable "image_id" {
2   type = string
3 }
4
5 variable "security_group_ids" {
6   description = "(Optional) - List of security group IDs."
7   type        = list(string)
8   default     = []
9 }
10
11 variable "pg_version" {
12   description = "PostgreSQL version"
13   type        = string
14   default     = "15"
15   validation {
16     condition = contains(["12", "12-1c", "13", "13-1c", "14", "14-1c", "15", "15-1c", "16"], var.pg_version)
17     error_message = "Allowed PostgreSQL versions are 12, 12-1c, 13, 13-1c, 14, 14-1c, 15, 15-1c, 16."
18   }
19 }
20
21 variable "access_policy" {
22   description = "Access policy from other services to the PostgreSQL cluster."
23   type = object({
24     data_lens    = optional(bool, null)
25     web_sql     = optional(bool, null)
26     serverless   = optional(bool, null)
27     data_transfer = optional(bool, null)
28   })
29   default = {}
30 }
31
32 variable "example" {
33   type      = string
34   nullable = false
35 }
36
37 variable "user_information" {
38   type = object({
39     name    = string
40     address = string
41   })
42   sensitive = true
43 }
44
45 resource "some_resource" "a" {
46   name    = var.user_information.name
47   address = var.user_information.address
48 }
```

## Способы задания переменных

Terraform загружает переменные в следующем порядке: 1 Переменные окружения. 2 Файл `terraform.tfvars`, если он есть. 3 Файл `terraform.tfvars.json`, если он есть. 4 Любые файлы `.auto.tfvars` или `.auto.tfvars.json`, обрабатываемые в алфавитном порядке их имён. 5 Любые опции `-var` и `-var-file` в командной строке в порядке их предоставления.

В командной строке

```

1 terraform apply -var="image_id=fd8ne6e3etbrr2ve9nlc"
2 terraform apply -var="image_id_list=[\"fd8ne6e3etbrr2ve9nlc\", \"fd8fsjddp35jvb4e4jo7\"]" -var="cores=4"
3 terraform apply -var='subnet_map={"ru-central1-a":"subnet-a", "ru-central1-b":"subnet-b"}'
```

В файле

```

1 terraform apply -var-file="testing.tfvars"
2
3 image_id = "fd8ne6e3etbrr2ve9nlc"
4 zone_names = [
5   "ru-central1-a",
6   "ru-central1-b",
7 ]
```

Переменные окружения Terraform ищет переменные окружения (начинающиеся с TF\_VAR\_), за которым следует имя объявленной переменной.

```
1 export TF_VAR_image_id=fd8ne6e3etbrr2ve9nlc
2 terraform plan
```

### Локальные переменные

```
1 locals {
2     boot_disk_name      = "${var.name_prefix}-boot-disk"
3     linux_vm_name       = "${var.name_prefix}-linux-vm"
4
5     labels = {
6         owner = "DevOps"
7         env   = "dev"
8     }
9 }
10
11 resource "yandex_compute_disk" "boot_disk" {
12     name = local.boot_disk_name
13
14 ...
15
16     labels = local.labels
17 }
18
19 resource "yandex_compute_instance" "this" {
20     name = local.linux_vm_name
21
22 ...
23
24     labels = local.labels
25 }
26
27 resource "yandex_vpc_subnet" "this" {
28 ...
29
30     labels = local.labels
31 }
```

### Выходные переменные/значения

```
1 output "boot_disk_id" {
2     description = "The ID of the boot disk created for the instance."
3     value        = yandex_compute_disk.boot_disk.id
4 }
5
6 output "instance_id" {
7     description = "The ID of the Yandex Compute instance."
8     value        = yandex_compute_instance.this.id
9 }
10
11 output "subnet_id" {
12     description = "The ID of the VPC subnet used by the Yandex Compute instance."
13     value        = yandex_vpc_subnet.private.id
14 }
15
16 output "ydb_id" {
17     description = "The ID of the Yandex Managed Service for YDB instance."
18     value        = yandex_ydb_database_serverless.this.id
19 }
20
21 output "service_account_id" {
22     description = "The ID of the Yandex IAM service account."
23     value        = yandex_iam_service_account.bucket.id
24 }
25
26 output "bucket_name" {
27     description = "The name of the Yandex Object Storage bucket."
28     value        = yandex_storage_bucket.this.bucket
29 }
30
31 output "access_key" {
32     description = "access key"
33     sensitive   = true
34     value        = yandex_iam_service_account_static_access_key.this
35 }
```

### Sensitive значения

```
1 > terraform output access_key
```

## 2.2.5 Функции

### Встроенные функции Terraform

Работа с числами - min, max, pow, floor, ceil  
 Работа со строками - format, join, split, substr  
 Работа с коллекциями - length, lookup, flatten, element, map, merge  
 Кодирование - base64decode, base64encode, jsonencode  
 Операции с файловой системой - file, fileexists, abspath, templatefile  
 Работа с датой и временем - formatdate, timestamp  
 Шифрование и хеш-функции - base64sha512, bcrypt  
 Работы с сетевой CIDR - cidrsubnet, cidrhost  
 Конвертирование типов - tobool, tomap, tolist

```

1  variable "value" {
2      default = "hello"
3  }
4
5  output "transformed_value" {
6      value = upper(format("transformed: %s", var.value))
7  }
```

```

1  terraform console
2
3  cidrnetmask("172.16.0.0/12")
```

## 2.2.6 Управляющие структуры

### Условные выражения

```

1  condition ? true_value : false_value
2
3  var.name == "" ? "default-name" : var.name
```

### Операторы

```

1  a == b
2  a != b
3  a < b
4  a <= b
5  a > b
6  a >= b
7  a || b
8  a && b
9  !a
```

### Циклы

```

1  [for s in var.list : upper(s)]
2
3  [for k, v in var.map : length(k) + length(v)]
```

```

1  variable "servers_count" {
2      default = {
3          "db" = 3,
4          "frontend" = 2,
5          "backend" = 5,
6          "balancer" = 1
7      }
8  }
9
10 output "servers" {
11     value = [for k, v in var.servers_count:
12         "${k} has ${v} servers"
13     ]
14 }
15
16 Outputs:
17
18 servers = [
19     "backend has 5 servers",
20     "balancer has 1 servers",
21     "db has 3 servers",
22     "frontend has 2 servers",
23 ]
```

Тип данных результата Скобки вокруг выражения for определяют, каким будет тип результирующего объекта:

```
1  квадратные скобки [ ] – tuple;
2  фигурные скобки { } – object.
```

### Фильтрация значений

```
1  [for s in var.list : upper(s) if s != ""]
```

## 2.2.7 Создание нескольких ресурсов

### Count



1 Если блок ресурса или модуля включает аргумент count, значение которого является целым числом. Terraform создаст указанное количество экземпляров.



1 ! Значение count должно быть известно до выполнения кода Terraform. Таким образом, нельзя использовать в качестве значения атрибуты, неизвестные до применения изменений.

```
1 resource "yandex_compute_instance" "this" {
2   count = 5
3
4   name      = "Server ${count.index}"
5   platform_id = "standard-v1"
6   zone      = "ru-central1-a"
7
8   resources {
9     cores  = 2
10    memory = 4
11  }
12
13   boot_disk {
14     initialize_params {
15       image_id = "fd833v6c5tb0udvk4jo6"
16     }
17   }
18
19   network_interface {
20     subnet_id = yandex_vpc_subnet.this.id
21   }
22 }
```

### For\_each

Если ваши экземпляры практически идентичны, использование count уместно. Если некоторые из аргументов нельзя задать с помощью целых чисел count.index, то безопаснее использовать for\_each. Он лишён недостатка count, поскольку

экземпляры в нём идентифицируются строкой. Аргумент `for_each` принимает типы данных `map(string)` или `set(string)` и создаёт экземпляр для каждого элемента этих типов данных.

```

1 variable "instances" {
2   type = map(object({
3     zone = string
4     resources = object({
5       cores = number
6       memory = number
7     })
8   }))
9   default = {
10    "server-1" = {
11      zone = "ru-central1-a"
12      resources = {
13        cores = 2
14        memory = 2
15      }
16    }
17    "server-2" = {
18      zone = "ru-central1-b"
19      resources = {
20        cores = 4
21        memory = 4
22      }
23    }
24  }
25 }
26
27 resource "yandex_compute_instance" "this" {
28   for_each = var.instances
29
30   name = each.key
31   zone = each.value["zone"]
32   resources {
33     cores = each.value.resources.cores
34     memory = each.value.resources.memory
35   }
36
37 ...
38 }
39 }
```

```

1 variable "databases" {
2   description = "List of PostgreSQL databases."
3
4   type = list(object({
5     name          = string
6     owner         = string
7     lc_collate   = optional(string, null)
8     lc_type       = optional(string, null)
9     template_db  = optional(string, null)
10    deletion_protection = optional(bool, null)
11    extensions   = optional(list(string), [])
12  }))
13 }
14
15 resource "yandex_mdb_postgresql_database" "database" {
16   for_each = length(var.databases) > 0 ? { for db in var.databases : db.name => db } : {}
17
18   name = each.value.name
19 ...
20 }
```

## 2.2.8 Динамические блоки

Некоторые типы ресурсов содержат повторяемые вложенные блоки в своих аргументах, обычно они представляют собой отдельные объекты. Вы можете динамически конструировать вложенные блоки с помощью dynamic-блоков. Они поддерживаются внутри блоков resource, data, provider и provisioner.

```

1 resource "yandex_vpc_security_group" "this" {
2 ...
3   ingress {
4     protocol      = "TCP"
5     description    = "HTTP"
6     v4_cidr_blocks = ["10.0.1.0/24"]
7     port          = 80
8   }
9
10  ingress {
11    protocol      = "TCP"
12    description    = "HTTPS"
13    v4_cidr_blocks = ["0.0.0.0/0"]
14    port          = 443
15  }
16 ...
17 }
```

```

1 variable "ingress_rules" {
2   type = list(object({
3     protocol      = string
4     description    = string
5     v4_cidr_blocks = list(string)
6     port          = number
7   }))
8   default = [
9     {
10       protocol      = "TCP"
11       description    = "HTTP"
12       v4_cidr_blocks = ["10.0.1.0/24"]
13       port          = 80
14     },
15     {
16       protocol      = "TCP"
17       description    = "HTTPS"
18       v4_cidr_blocks = ["0.0.0.0/0"]
19       port          = 443
20     }
21   ]
22 }
23
24 resource "yandex_vpc_security_group" "this"
25 ...
26 dynamic "ingress" {
27   for_each = var.ingress_rules
28   content {
29     protocol      = ingress.value.protocol
30     description    = ingress.value.description
31     port          = ingress.value.port
32     v4_cidr_blocks = ingress.value.v4_cidr_blocks
33   }
34 ...
35 }
```

```

1 dynamic "network_interface" {
2   for_each = var.network_interfaces
3   content {
4     subnet_id      = network_interface.value.ubnet_id
5     nat            = network_interface.value.nat
6     nat_ip_address = network_interface.value.nat_ip_address
7     security_group_ids = network_interface.value.security_group_ids
8     ip_address      = network_interface.value.ip_address
9
10    dynamic "dns_record" {
11      for_each = network_interface.value.dns_record
12      content {
13        fqdn          = dns_record.value.fqdn
14        dns_zone_id  = dns_record.value.dns_zone_id
15        ttl           = dns_record.value.ttl
16        ptr           = dns_record.value.ptr
17      }
18    }
19  }
20 }
```

## 2.2.9 Указание зависимости

В большинстве случаев Terraform сам управляет зависимостями. Вам нужно явно указать зависимость только в том случае, когда ресурс или модуль зависит от поведения другого ресурса, но не обращается к данным этого ресурса в своих аргументах.

```

1  variable "databases" {
2  ...
3    type = list(object({
4      name = string
5    }))
6  ...
7 }
8
9  variable "users" {
10 ...
11   type = list(object({
12     name          = string
13     password      = optional(string, null)
14     authentication_plugin = optional(string, null)
15     global_permissions = optional(list(string), [])
16     connection_limits = optional(object({
17       max_questions_per_hour = optional(number, -1)
18       max_updates_per_hour   = optional(number, -1)
19       max_connections_per_hour = optional(number, -1)
20       max_user_connections   = optional(number, -1)
21     }), null)
22     permissions = optional(list(object({
23       database_name = string
24       roles         = optional(list(string), ["ALL"])
25     })), []
26   }))
27 ...
28 }
29
30 resource "yandex_mdb_mysql_database" "database" {
31   for_each = length(var.databases) > 0 ? { for db in var.databases : db.name => db } : {}
32
33   cluster_id = yandex_mdb_mysql_cluster.this.id
34   name        = lookup(each.value, "name", null)
35 }
36
37 resource "yandex_mdb_mysql_user" "user" {
38 ...
39
40   dynamic "permission" {
41     for_each = lookup(each.value, "permissions", [])
42     content {
43       database_name = permission.value.database_name
44       roles         = permission.value.roles
45     }
46   }
47 ...
48   # Create databases before users
49   depends_on = [
50     yandex_mdb_mysql_database.database
51   ]
52 }
```

## 2.2.10 State-файл в Terraform

Изменения в файле backend.tf требуют повторной инициализации

Решение: используйте команду `terraform init -reconfigure`.

В момент применения конфигурации пропала связь с бэкендом

Решение: исправьте командой `terraform force-unlock LOCK_ID`.

Пример `LOCK_ID`: `yandex_storage_bucket.dev`.

### Блоке import

Блок `import` — это новый декларативный способ описывать импорт ресурсов прямо в Terraform-коде, без ручного запуска `terraform import ...` в консоли.

- ◆ Проще говоря: ты описываешь в коде, какие ресурсы нужно импортировать в state, и Terraform сам делает импорт при terraform apply.

```
1 import {
2     to = yandex_compute_instance.testvm
3     id = "<< VM ID >>"
4 }
5
6 resource "yandex_compute_instance" "testvm" {
7     name          = "testvm"
8     platform_id   = "standard-v3"
9     allow_stopping_for_update = true
10
11    resources {
12        cores = 2
13        memory = 2
14    }
15
16    scheduling_policy {
17        preemptible = true
18    }
19
20    boot_disk {
21        disk_id = yandex_compute_disk.testvm.id
22        auto_delete = false
23    }
24
25    network_interface {
26        subnet_id = yandex_vpc_subnet.dev.id
27        nat       = true
28    }
29
30    metadata = {
31        enable-oslogin = true
32    }
33 }
```

2.2.11 Блокировку состояния инфраструктуры

---

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94