

# **Техническое решение аналитического сервера для работы аналитиков**

## **1. Архитектура и необходимое ПО**

### **1.1 Основные компоненты системы**

#### **Операционная система**

- **Ubuntu Server 22.04 LTS** - стабильная серверная ОС с долгосрочной поддержкой

#### **База данных**

- **PostgreSQL 15** - основная реляционная БД для хранения данных
- **Redis 7** - кеширование и очереди задач
- **ClickHouse** (опционально) - для больших объемов аналитических данных

#### **Система контроля версий**

- **GitLab Community Edition** - внутренний Git-сервер с веб-интерфейсом
- **Git LFS** - для работы с большими файлами данных

#### **Оркестрация и планировщик задач**

- **Apache Airflow 2.7** - управление workflows и автоматизация
- **Celery** - распределенная очередь задач (входит в Airflow)

#### **Python-окружение**

- **Python 3.10+** - основной язык для анализа
- **Conda/Miniconda** - управление пакетами и окружениями
- **JupyterHub** - многопользовательские Jupyter notebooks
- **Основные пакеты:** pandas, numpy, scikit-learn, matplotlib, seaborn, plotly

#### **Веб-сервер и прокси**

- **Nginx** - обратный прокси и веб-сервер
- **Gunicorn** - WSGI-сервер для Python приложений

#### **Безопасность и управление секретами**

- **HashiCorp Vault** - управление паролями и секретами
- **Let's Encrypt** - SSL-сертификаты
- **UFW (Uncomplicated Firewall)** - базовый файрволл

## **Мониторинг и логирование**

- **Prometheus + Grafana** - мониторинг системы
- **ELK Stack** (Elasticsearch, Logstash, Kibana) - централизованное логирование
- **Fail2ban** - защита от брутфорса

## **Дополнительные инструменты**

- **Docker + Docker Compose** - контейнеризация сервисов
- **Portainer** - веб-интерфейс для управления Docker
- **Backup tools:** pg\_dump, rsync, restic

## **2. Схема архитектуры сервера**



### 3. Рекомендуемые системные требования

#### Минимальная конфигурация:

- **CPU:** 8 cores (Intel Xeon или AMD EPYC)
- **RAM:** 32 GB

- **Storage:** 1TB SSD (система) + 4TB HDD (данные)
- **Network:** 1 Gbps

## Рекомендуемая конфигурация:

- **CPU:** 16-24 cores
- **RAM:** 64-128 GB
- **Storage:** 2TB NVMe SSD (система) + 8TB SAS HDD (данные)
- **Network:** 10 Gbps

## 4. Пошаговая инструкция по установке и настройке

### Шаг 1: Подготовка системы

```
bash

# Обновление системы
sudo apt update && sudo apt upgrade -y

# Установка базовых пакетов
sudo apt install -y curl wget git vim htop tree unzip software-properties-common

# Настройка часового пояса
sudo timedatectl set-timezone Europe/Moscow

# Создание пользователя для сервисов
sudo useradd -m -s /bin/bash analytics
sudo usermod -aG sudo analytics
```

### Шаг 2: Установка Docker и Docker Compose

```
bash
```

```
# Установка Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Добавление пользователя в группу docker
sudo usermod -aG docker $USER
sudo usermod -aG docker analytics

# Установка Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Перезагрузка для применения изменений
sudo reboot
```

## Шаг 3: Установка PostgreSQL

```
bash

# Установка PostgreSQL 15
sudo apt install -y postgresql-15 postgresql-contrib-15

# Настройка PostgreSQL
sudo -u postgres psql -c "ALTER USER postgres PASSWORD 'your_secure_password';"

# Создание базы данных для Airflow
sudo -u postgres createdb airflow_db
sudo -u postgres psql -c "CREATE USER airflow WITH PASSWORD 'airflow_password';"
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE airflow_db TO airflow;"

# Создание базы данных для аналитики
sudo -u postgres createdb analytics_db
sudo -u postgres psql -c "CREATE USER analytics WITH PASSWORD 'analytics_password';"
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE analytics_db TO analytics;"
```

## Шаг 4: Установка Redis

```
bash
```

```
# Установка Redis
sudo apt install -y redis-server

# Настройка Redis
sudo systemctl enable redis-server
sudo systemctl start redis-server

# Проверка статуса
sudo systemctl status redis-server
```

## Шаг 5: Установка Python и Conda

```
bash

# Установка Miniconda
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh -b -p /opt/miniconda3

# Настройка PATH для всех пользователей
echo 'export PATH="/opt/miniconda3/bin:$PATH"' | sudo tee -a /etc/profile

# Создание окружения для аналитики
source /opt/miniconda3/bin/activate
conda create -n analytics python=3.10 -y
conda activate analytics

# Установка основных пакетов
conda install -y pandas numpy scikit-learn matplotlib seaborn jupyter notebook ipykernel
pip install plotly dash streamlit apache-airflow[postgres,redis]==2.7.0
```

## Шаг 6: Установка и настройка GitLab CE

```
bash
```

```
# Установка зависимостей
sudo apt install -y curl openssh-server ca-certificates tzdata perl

# Добавление репозитория GitLab
curl -sS https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh | sudo bash

# Установка GitLab CE
sudo EXTERNAL_URL="https://git.yourdomain.com" apt install gitlab-ce

# Первоначальная настройка
sudo gitlab-ctl reconfigure

# Получение начального пароля root
sudo cat /etc/gitlab/initial_root_password
```

## Шаг 7: Установка и настройка Apache Airflow

bash

```
# Создание директории для Airflow
sudo mkdir -p /opt/airflow
sudo chown analytics:analytics /opt/airflow

# Переключение на пользователя analytics
sudo su - analytics

# Настройка переменных окружения
export AIRFLOW_HOME=/opt/airflow
echo 'export AIRFLOW_HOME=/opt/airflow' >> ~/.bashrc

# Активация conda окружения
source /opt/miniconda3/bin/activate analytics

# Инициализация базы данных Airflow
airflow db init

# Создание админ пользователя
airflow users create \
... --username admin \
... --firstname Admin \
... --lastname User \
... --role Admin \
... --email admin@yourdomain.com \
... --password admin_password

# Создание systemd сервисов для Airflow
sudo tee /etc/systemd/system/airflow-webserver.service > /dev/null <<EOF
[Unit]
Description=Airflow webserver daemon
After=network.target postgresql.service mysql.service redis.service rabbitmq-server.service
Wants=postgresql.service mysql.service redis.service rabbitmq-server.service

[Service]
Environment=AIRFLOW_HOME=/opt/airflow
User=analytics
Group=analytics
Type=notify
ExecStart=/opt/miniconda3/envs/analytics/bin/airflow webserver --port 8080
Restart=on-failure
RestartSec=5s
PrivateTmp=true

[Install]
WantedBy=multi-user.target
EOF
```

```
sudo tee /etc/systemd/system/airflow-scheduler.service > /dev/null <<EOF
[Unit]
Description=Airflow scheduler daemon
After=network.target postgresql.service mysql.service redis.service rabbitmq-server.service
Wants=postgresql.service mysql.service redis.service rabbitmq-server.service

[Service]
Environment=AIRFLOW_HOME=/opt/airflow
User=analytics
Group=analytics
Type=notify
ExecStart=/opt/miniconda3/envs/analytics/bin/airflow scheduler
Restart=always
RestartSec=5s

[Install]
WantedBy=multi-user.target
EOF

# Включение и запуск сервисов
sudo systemctl daemon-reload
sudo systemctl enable airflow-webserver airflow-scheduler
sudo systemctl start airflow-webserver airflow-scheduler
```

## Шаг 8: Установка JupyterHub

bash

```
# Установка Node.js для JupyterHub
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs

# Установка JupyterHub
sudo /opt/miniconda3/envs/analytics/bin/pip install jupyterhub jupyterlab

# Создание конфигурационного файла
sudo mkdir -p /etc/jupyterhub
sudo /opt/miniconda3/envs/analytics/bin/jupyterhub --generate-config -f /etc/jupyterhub/jupyterhub_config.py

# Настройка JupyterHub
sudo tee -a /etc/jupyterhub/jupyterhub_config.py > /dev/null <<EOF
c.JupyterHub.bind_url = 'http://127.0.0.1:8000'
c.JupyterHub.hub_bind_url = 'http://127.0.0.1:8081'
c.Spawner.default_url = '/lab'
c.Spawner.cmd = ['/opt/miniconda3/envs/analytics/bin/jupyter-labhub']
EOF

# Создание systemd сервиса для JupyterHub
sudo tee /etc/systemd/system/jupyterhub.service > /dev/null <<EOF
[Unit]
Description=JupyterHub
After=syslog.target network.target

[Service]
User=root
Environment="PATH=/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/opt/miniconda3/envs/analytics/bin"
ExecStart=/opt/miniconda3/envs/analytics/bin/jupyterhub -f /etc/jupyterhub/jupyterhub_config.py

[Install]
WantedBy=multi-user.target
EOF

sudo systemctl daemon-reload
sudo systemctl enable jupyterhub
sudo systemctl start jupyterhub
```

## Шаг 9: Установка HashiCorp Vault

bash

```
# Добавление репозитория HashiCorp
wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update && sudo apt install vault

# Создание конфигурационного файла
sudo mkdir -p /etc/vault.d
sudo tee /etc/vault.d/vault.hcl > /dev/null <<EOF
storage "file" {
  . path = "/opt/vault/data"
}

listener "tcp" {
  . address = "127.0.0.1:8200"
  . tls_disable = 1
}

ui = true
EOF

# Создание директории для данных
sudo mkdir -p /opt/vault/data
sudo chown vault:vault /opt/vault/data

# Создание systemd сервиса
sudo tee /etc/systemd/system/vault.service > /dev/null <<EOF
[Unit]
Description=Vault secret management tool
After=network.target

[Service]
User=vault
Group=vault
ExecStart=/usr/bin/vault server -config=/etc/vault.d/vault.hcl
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF

sudo systemctl daemon-reload
```

```
sudo systemctl enable vault
```

```
sudo systemctl start vault
```

## Шаг 10: Установка и настройка Nginx

bash

```
# Установка Nginx
sudo apt install -y nginx

# Создание конфигурации для обратного прокси
sudo tee /etc/nginx/sites-available/analytics-server > /dev/null <<EOF
server {
    listen 80;
    server_name yourdomain.com;

    # GitLab
    location /gitlab/ {
        proxy_pass http://127.0.0.1:80/;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
    }

    # JupyterHub
    location /jupyter/ {
        proxy_pass http://127.0.0.1:8000/;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;

        # WebSocket support
        proxy_http_version 1.1;
        proxy_set_header Upgrade \$http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    # Airflow
    location /airflow/ {
        proxy_pass http://127.0.0.1:8080/;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
    }
}

# Активация конфигурации
sudo ln -s /etc/nginx/sites-available/analytics-server /etc/nginx/sites-enabled/
sudo rm /etc/nginx/sites-enabled/default
```

```
# Проверка конфигурации и перезапуск
```

```
sudo nginx -t
```

```
sudo systemctl enable nginx
```

```
sudo systemctl restart nginx
```

## Шаг 11: Настройка мониторинга (Prometheus + Grafana)

bash

```
# Создание Docker Compose файла для мониторинга
mkdir -p /opt/monitoring
cd /opt/monitoring
```

```
tee docker-compose.yml > /dev/null <<EOF
```

```
version: '3.8'
```

```
services:
```

```
  prometheus:
```

```
    image: prom/prometheus:latest
```

```
    container_name: prometheus
```

```
    ports:
```

```
      - "9090:9090"
```

```
    volumes:
```

```
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
```

```
      - prometheus-data:/prometheus
```

```
    command:
```

```
      - '--config.file=/etc/prometheus/prometheus.yml'
```

```
      - '--storage.tsdb.path=/prometheus'
```

```
      - '--web.console.libraries=/etc/prometheus/console_libraries'
```

```
      - '--web.console.templates=/etc/prometheus/consoles'
```

```
  grafana:
```

```
    image: grafana/grafana:latest
```

```
    container_name: grafana
```

```
    ports:
```

```
      - "3000:3000"
```

```
    volumes:
```

```
      - grafana-data:/var/lib/grafana
```

```
    environment:
```

```
      - GF_SECURITY_ADMIN_PASSWORD=admin_password
```

```
  node-exporter:
```

```
    image: prom/node-exporter:latest
```

```
    container_name: node-exporter
```

```
    ports:
```

```
      - "9100:9100"
```

```
volumes:
```

```
  prometheus-data:
```

```
  grafana-data:
```

```
EOF
```

```
# Создание конфигурации Prometheus
```

```
tee prometheus.yml > /dev/null <<EOF
```

```
global:
```

```
scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'node-exporter'
    static_configs:
      - targets: ['node-exporter:9100']

  - job_name: 'airflow'
    static_configs:
      - targets: ['host.docker.internal:8080']

EOF

# Запуск мониторинга
docker-compose up -d
```

## Шаг 12: Настройка безопасности

bash

```
# Установка и настройка UFW
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow ssh
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw --force enable

# Установка Fail2ban
sudo apt install -y fail2ban

# Настройка Fail2ban
sudo tee /etc/fail2ban/jail.local > /dev/null <<EOF
[DEFAULT]
bantime = 3600
findtime = 600
maxretry = 3

[sshd]
enabled = true
port = ssh
logpath = /var/log/auth.log

[nginx-http-auth]
enabled = true
port = http,https
logpath = /var/log/nginx/error.log
EOF

sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```

## Шаг 13: Настройка SSL (Let's Encrypt)

```
bash

# Установка Certbot
sudo apt install -y certbot python3-certbot-nginx

# Получение SSL сертификата
sudo certbot --nginx -d yourdomain.com

# Автоматическое обновление сертификатов
sudo crontab -e
# Добавить строку: 0 12 * * * /usr/bin/certbot renew --quiet
```

## Шаг 14: Настройка системы резервного копирования

```
bash
```

```
# Создание скрипта резервного копирования
sudo tee /opt/backup.sh > /dev/null <<EOF
#!/bin/bash
BACKUP_DIR="/backup"
DATE=$(date +\%Y\%m\%d_\%H\%M\%S)

# Создание директории для бэкапов
mkdir -p \$BACKUP_DIR

# Бэкап PostgreSQL
sudo -u postgres pg_dump airflow_db > \$BACKUP_DIR/airflow_db_\$DATE.sql
sudo -u postgres pg_dump analytics_db > \$BACKUP_DIR/analytics_db_\$DATE.sql

# Бэкап GitLab
sudo gitlab-backup create

# Бэкап конфигураций
tar -czf \$BACKUP_DIR/configs_\$DATE.tar.gz /etc/nginx /etc/vault.d /opt/airflow/airflow.cfg

# Удаление старых бэкапов (старше 30 дней)
find \$BACKUP_DIR -name "*.sql" -mtime +30 -delete
find \$BACKUP_DIR -name "*.tar.gz" -mtime +30 -delete

echo "Backup completed: \$DATE"
EOF

sudo chmod +x /opt/backup.sh

# Добавление в cron для ежедневного выполнения
sudo crontab -e
# Добавить строку: 0 2 * * * /opt/backup.sh >> /var/log/backup.log 2>&1
```

## 5. Настройка рабочего места аналитика

### 5.1 Подключение к серверу через VSCode

1. Установить расширение "Remote - SSH" в VSCode
2. Добавить конфигурацию SSH в `~/.ssh/config`:

```
Host analytics-server
  HostName your-server-ip
  ....User your-username
  ....Port 22
  IdentityFile ~/.ssh/id_rsa
```

3. Подключиться через Command Palette: "Remote-SSH: Connect to Host"

## 5.2 Настройка Python окружения в VSCode

1. Установить расширение "Python" в VSCode
2. Выбрать интерпретатор: `/opt/miniconda3/envs/Analytics/bin/python`
3. Настроить `.vscode/settings.json` в проекте:

```
json

{
  .... "python.defaultInterpreterPath": "/opt/miniconda3/envs/Analytics/bin/python",
  .... "python.terminal.activateEnvironment": true
}
```

## 6. Использование системы

### 6.1 Работа с Git

- Клонирование репозитория: `git clone https://git.yourdomain.com/Analytics/project.git`
- Создание веток для фичей: `git checkout -b feature/new-analysis`
- Коммиты и пуши: стандартный Git workflow

### 6.2 Работа с Airflow

- Создание DAG'ов в директории `/opt/airflow/dags/`
- Мониторинг выполнения через веб-интерфейс
- Использование переменных и connections для конфигурации

### 6.3 Управление секретами через Vault

- Инициализация Vault: `vault operator init`
- Сохранение секретов: `vault kv put secret/db password=mypassword`
- Использование в скриптах через API или CLI

### 6.4 Работа с JupyterHub

- Доступ через веб-интерфейс

- Создание и обмен notebook'ами между пользователями
- Использование общих библиотек и данных

## 7. Обслуживание и мониторинг

### 7.1 Регулярные задачи

- Проверка логов: `journalctl -u airflow-scheduler -f`
- Мониторинг ресурсов через Grafana
- Обновление системы и пакетов
- Проверка резервных копий

### 7.2 Устранение неполадок

- Проверка статуса сервисов: `systemctl status service-name`
- Анализ логов приложений
- Мониторинг дискового пространства
- Проверка подключений к базам данных

## 8. Масштабирование

При росте нагрузки рекомендуется:

- Разделение сервисов на отдельные серверы
- Использование кластера Kubernetes
- Настройка репликации баз данных
- Внедрение системы распределенного хранения данных

Эта архитектура обеспечивает надежную, безопасную и масштабируемую среду для работы команды аналитиков с возможностью совместной разработки, автоматизации и мониторинга процессов.