МОСКВА
11-12 мая 2012

Application Developer Days
/*Программисты всех платформ, общайтесь!*/

# Разработка и администрирование через тестирование - облачный сервис «Битрикс24» в Амазоне

Александр Сербул
1С-Битрикс
@AlexSerbul

# Наши цели

- Посмотреть на  Amazon Web Services (AWS) с «птичьего полета»
- Научиться эффективно с ним работать, в т. ч. на PHP
- Уверенно развивать функционал, не отставая от рынка
- Держать созданную систему под постоянным контролем
- Изучить результат -  сервис «Битрикс24» - изнутри

# AWS с «птичьего полета»

*Управляемый из кода хостинг с множеством реализованных Enterprise-паттернов :*

- **Amazon Elastic Compute Cloud (EC2)** – серверы и образы
- **Amazon Elastic Block Store (EBS)** - внешние диски
- **Auto Scaling** – масштабирование (от нагрузки и т. д.)
- **Elastic Load Balancing** - балансировщики
- **Amazon Simple Storage Service (S3)** – облачное хранилище

# AWS с «птичьего полета»

- **Amazon Simple Queue Service (SQS)** - очереди сообщений
- **Amazon Simple Notification Service (SNS)** - уведомлялка
- **Amazon Simple Email Service (SES)** – рассылки почты
- **AWS Identity and Access Management (IAM)** – управление учетками

# AWS с «птичьего полета»

- **Amazon CloudFront** – CDN по всему миру
- **Amazon Route 53** – управление DNS

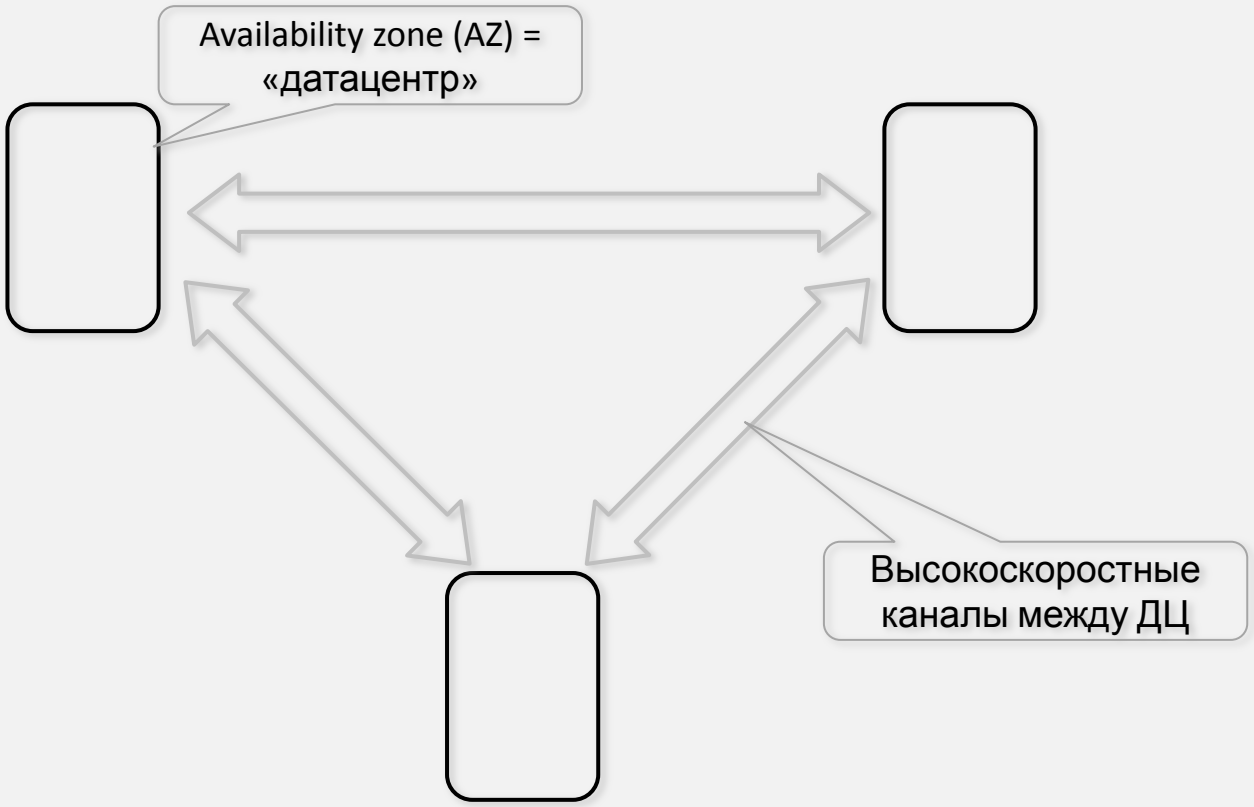
- **Amazon Relational Database Service (RDS)** – MySQL с slaves и auto-failover
- **Amazon DynamoDB, Amazon SimpleDB** - NoSQL
- **Amazon ElastiCache** - «неубиваемый» memcached

# AWS с «птичьего полета»

- **Amazon CloudWatch** – мониторинг + аналитика
- **Amazon Virtual Private Cloud (VPC)** – собственная подсетка (с кластерами)

Полезных сервисов много, готовые паттерны позволяют просто масштабироваться и обеспечивать неплохую отказоустойчивость (SLA до 99,95%).

# AWS с «птичьего полета»




Region = группа связанных датацентров

# AWS с «птичьего полета»

Availability zone (AZ) = «датацентр»

Серверы (EC2)
Диски (EBS) – доступны только внутри датацентра

Снепшоты дисков
Образы серверов
Данные в облаке (S3)
Учетки (IAM)
Балансировщики и др. – реплицируются «между» датацентрами, на уровне региона

Region = группа связанных датацентров

# AWS – все не так гладко

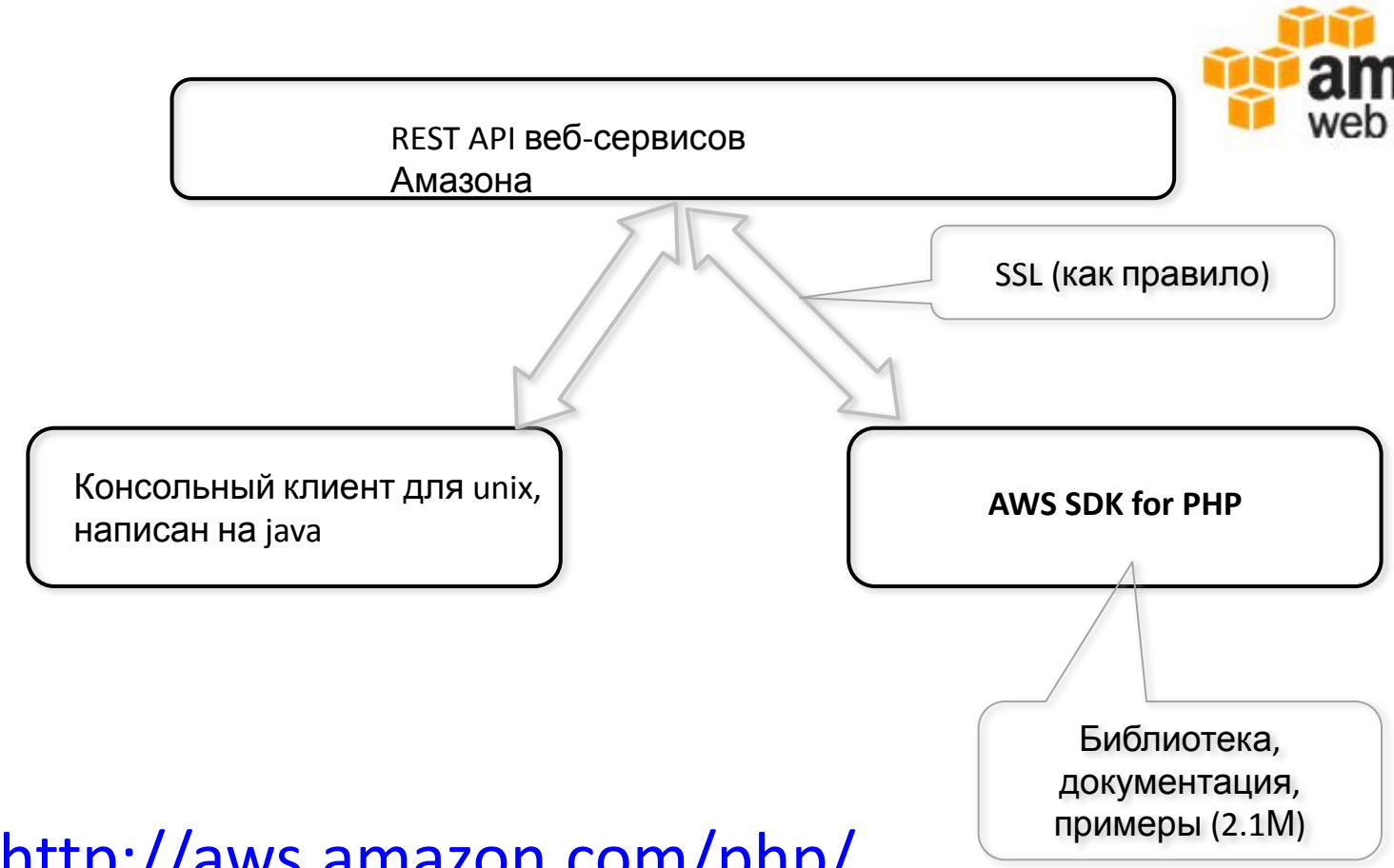
- Веб-сервисы внутри архитектурно отличаются друг от друга, в т.ч . по подходу и API. Не всегда удобно стыкуются.
- Прослеживается эволюция архитектур – например «наслоения концепций» в IAM/S3
- Имеются неочевидные «жесткие» ограничения и лимиты – например число учеток в IAM (<=5000)

# AWS – нужно строить свое управляющее ядро

- Внутренняя система мониторинга CloudWatch - пока очень ограничена, уступает напр. Nagios/Munin
- Измерения внутри машин (top) и из CloudWatch – могут отличаться в разы (CPU, LA, др.)
- Нередко «соседи» на виртуальной машине внезапно влияют на производительность (CPU Stolen %)

Нужно научиться держать конфигурацию под контролем, автоматизировав максимум операций.

# AWS SDK for PHP




- http://aws.amazon.com/php/
- http://aws.amazon.com/sdkforphp/

# AWS SDK for PHP - документация

- AmazonAS
- AmazonCloudFormation
- AmazonCloudFront
- AmazonCloudSearch
- AmazonCloudWatch
- AmazonDynamoDB
- AmazonEC2
- AmazonELB
- AmazonEMR
- AmazonElastiCache
- AmazonElasticBeanstalk
- AmazonIAM
- AmazonImportExport
- AmazonRDS
- AmazonS3
- AmazonSDB
- AmazonSES
- AmazonSNS
- AmazonSQS
- AmazonSTS
- AmazonSWF
- AmazonStorageGateway
- CacheAPC
- CacheMC
- CachePDO
- CacheXCache
- Core Utilities
- Extensions

CONTRIBUTORS
LICENSE
NOTICE

Method

## AmazonEC2

services/ec2.class.php

**run_instances** ( $image_id, $min_count, $max_count, $opt )

The RunInstances operation launches a specified number of instances.

If Amazon EC2 cannot launch the minimum number AMIs you request, no instances launch. If there is insufficient capacity to launch the maximum number of AMIs you request, Amazon EC2 launches as many as possible to satisfy the requested maximum values.

Every instance is launched in a security group. If you do not specify a security group at launch,

...re information on creating security

**Examples**

**Launch an EC2 instance.**

...on about instance types, see Instance

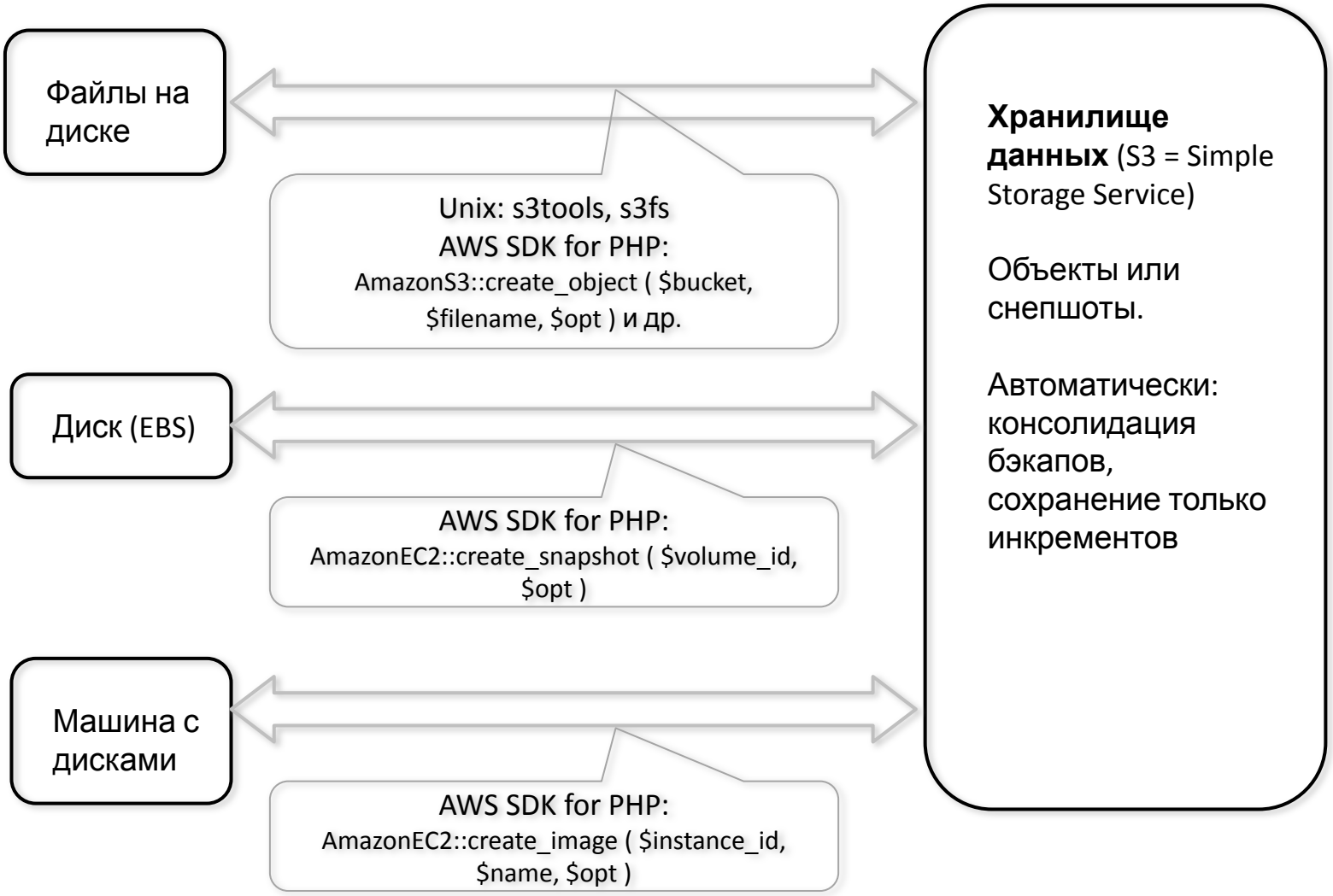
```php
// Instantiate the class
$ec2 = new AmazonEC2();

// Boot an instance of the image
$response = $ec2->run_instances('ami-84db39ed', 1, 1, array(
    'InstanceType' => 'm1.small'
));

// Success?
var_dump($response->isOK());
```

**Result:**

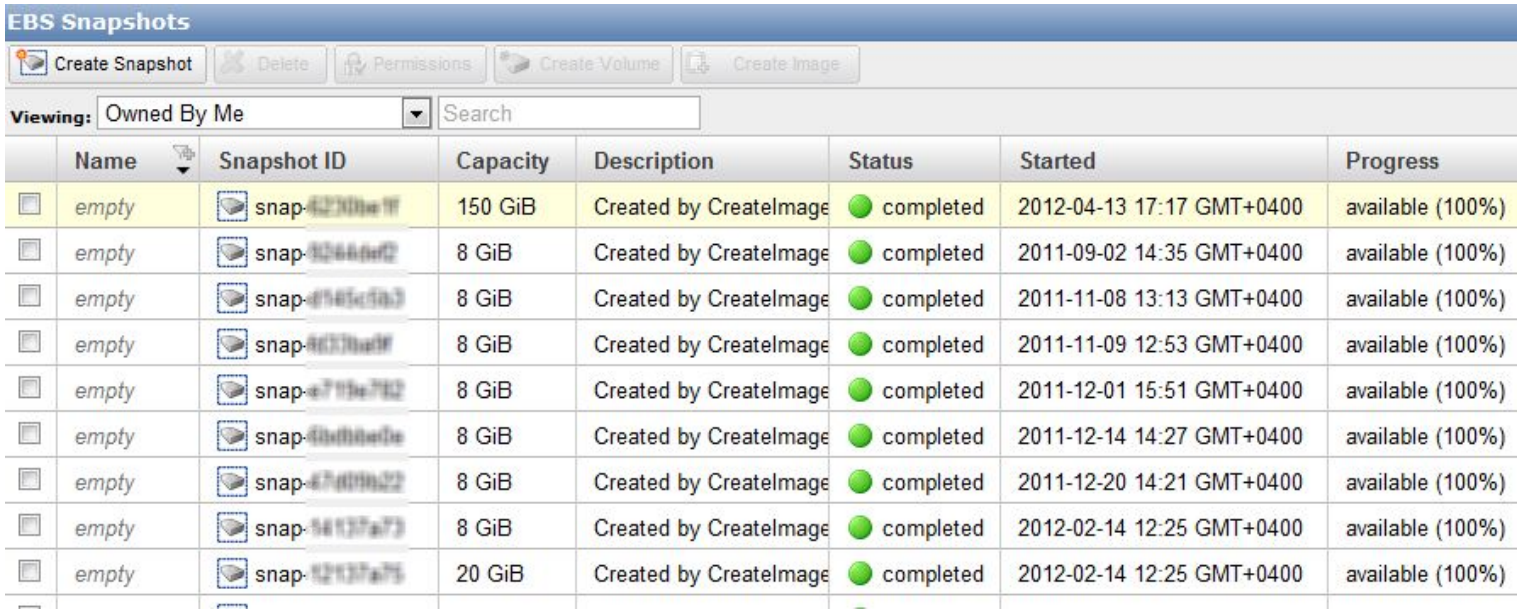
```
bool(true)
```

http://docs.amazonwebservices.com/AWSSDKforPHP/latest/

# Бэкап данных в S3/Snapshots



# Бэкап данных в S3/Snapshots



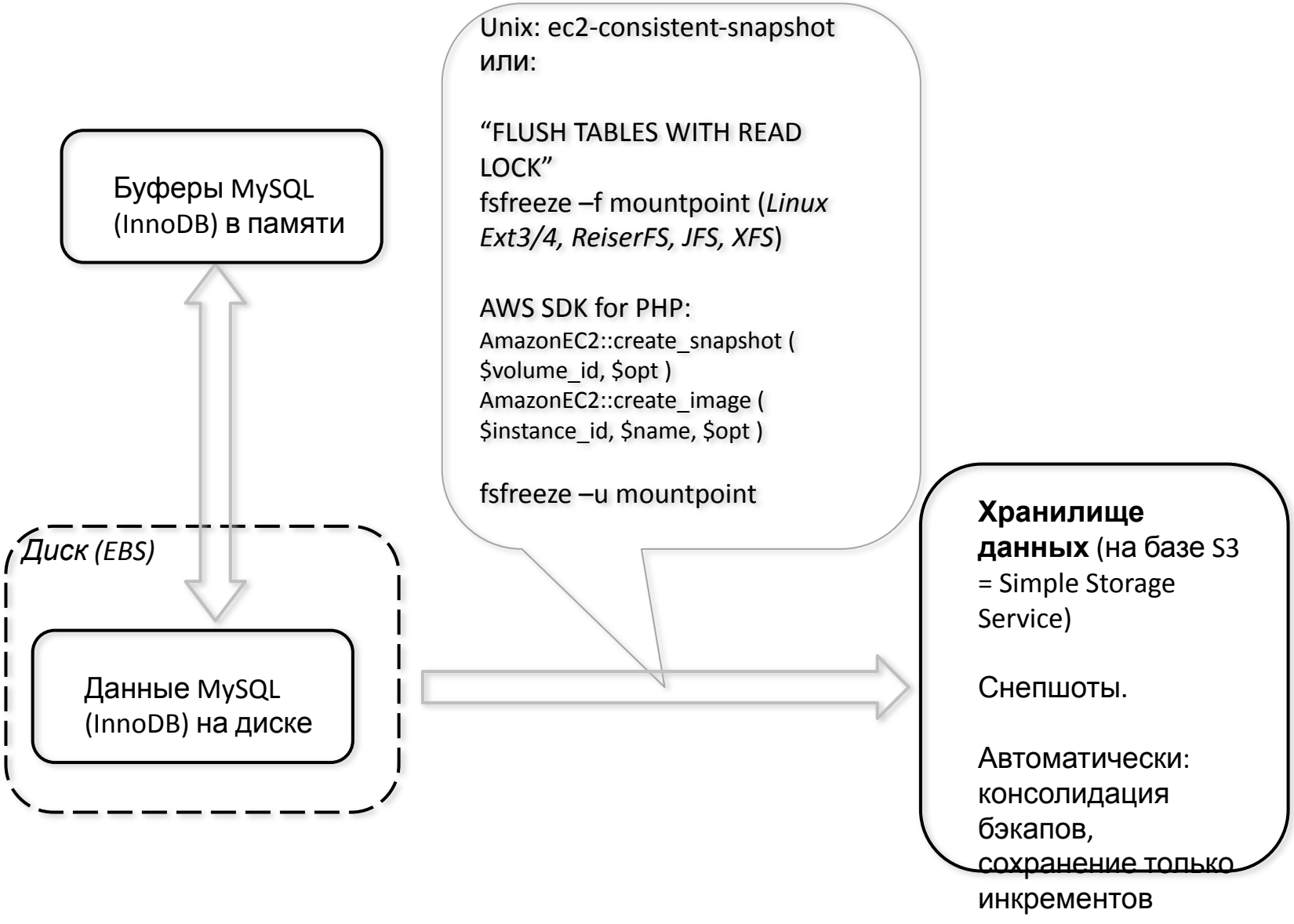

- Нет инструментов очистки устаревших снепшотов и образов машин, их нужно писать (иначе можно упереться в лимит).
- Нужно писать оболочку, для повтора неудавшихся операций, логирования и т.п.

# Бэкап данных в S3/Snapshots

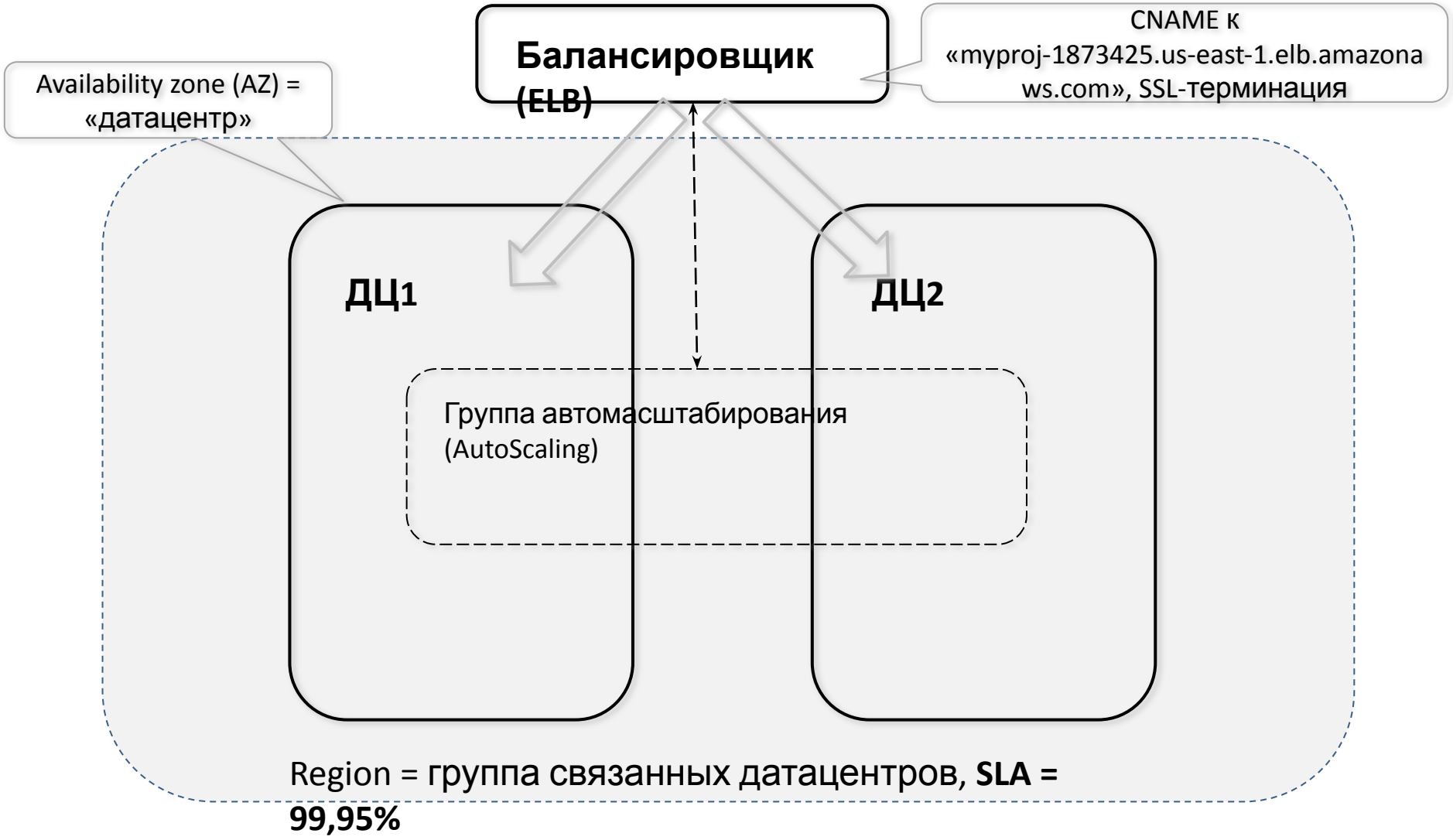
```
...
foreach ($vols as $path => $id) {

    $response = $ec2->create_snapshot($id, array('Description'=>"Snapshot:".$instanceRole.":".$path));

    if ( $response->isOK() ) {

      bxc_log("Started snapshot for ($id): ".$response->body->snapshotId);

      $r = $ec2->create_tags($response->body->snapshotId, array(
        array('Key' => 'Name', 'Value' => "Snapshot:".$instanceRole.":".$path),
        array('Key' => 'Role', 'Value' => "Snapshot:".$instanceRole.":".$path),
      ));

      if ( !$r->isOK() ) {
        bxc_log_amazon_error($response, __FUNCTION__.":".__LINE__);

      }

    } else {
      bxc_log_amazon_error($response, __FUNCTION__.":".__LINE__);
      return false;
    }
  }
```

- Полезно создаваемым объектам присваивать тэги, для дальнейшей фильтрации

# Бэкап MySQL в S3/Snapshots

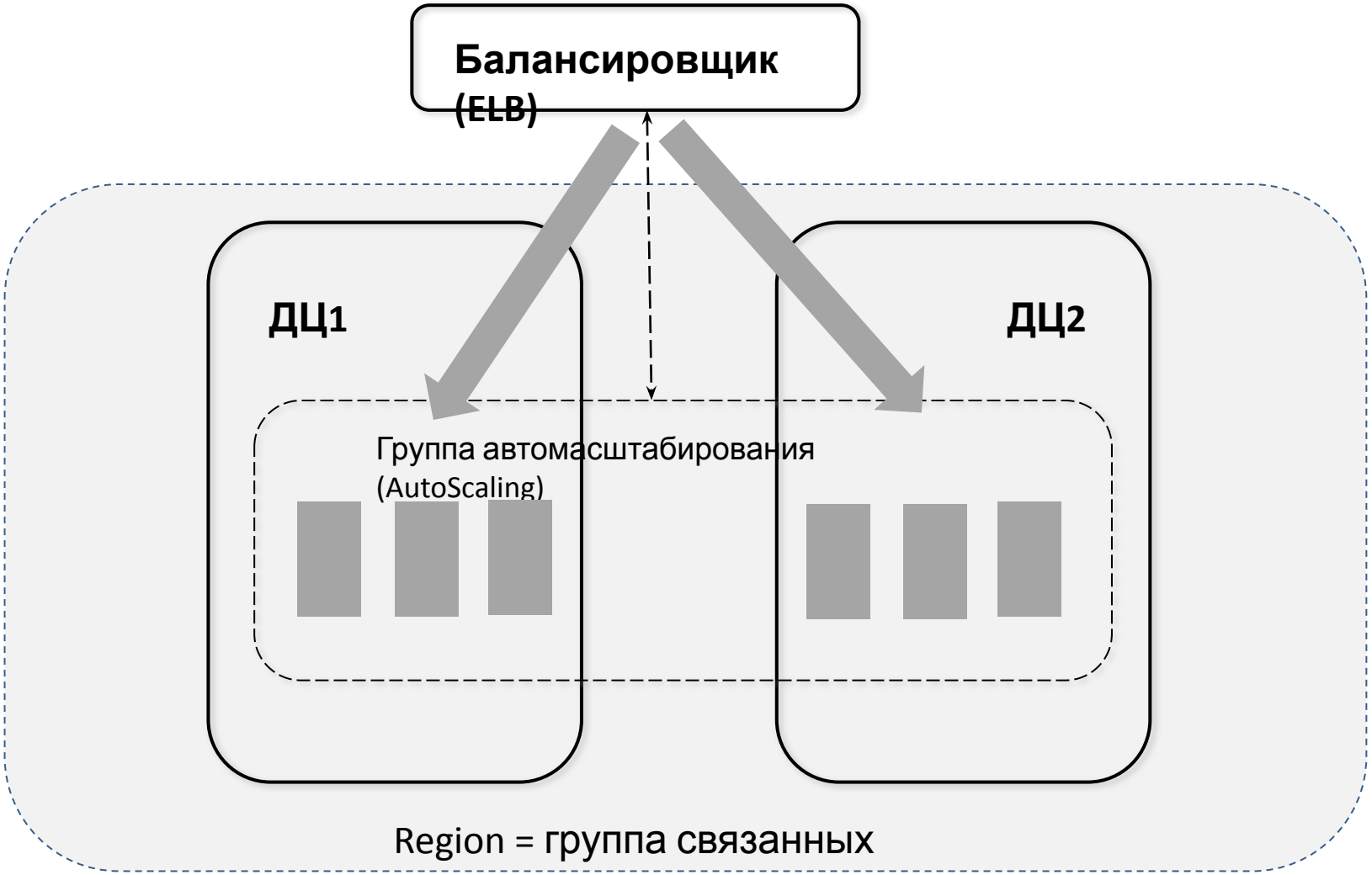


# Балансировка/Переключение трафика

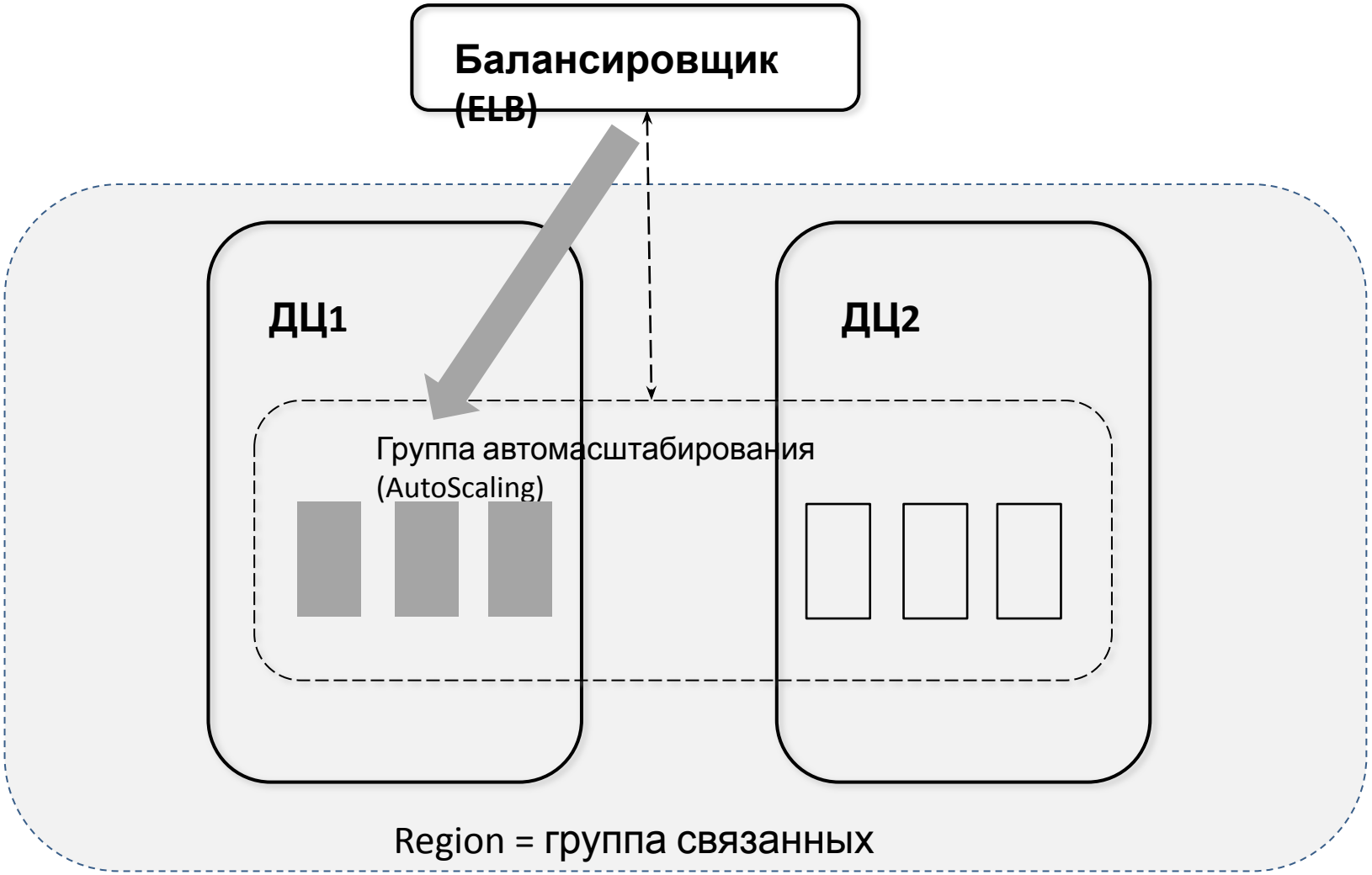



Region = группа связанных датацентров, **SLA = 99,95%**

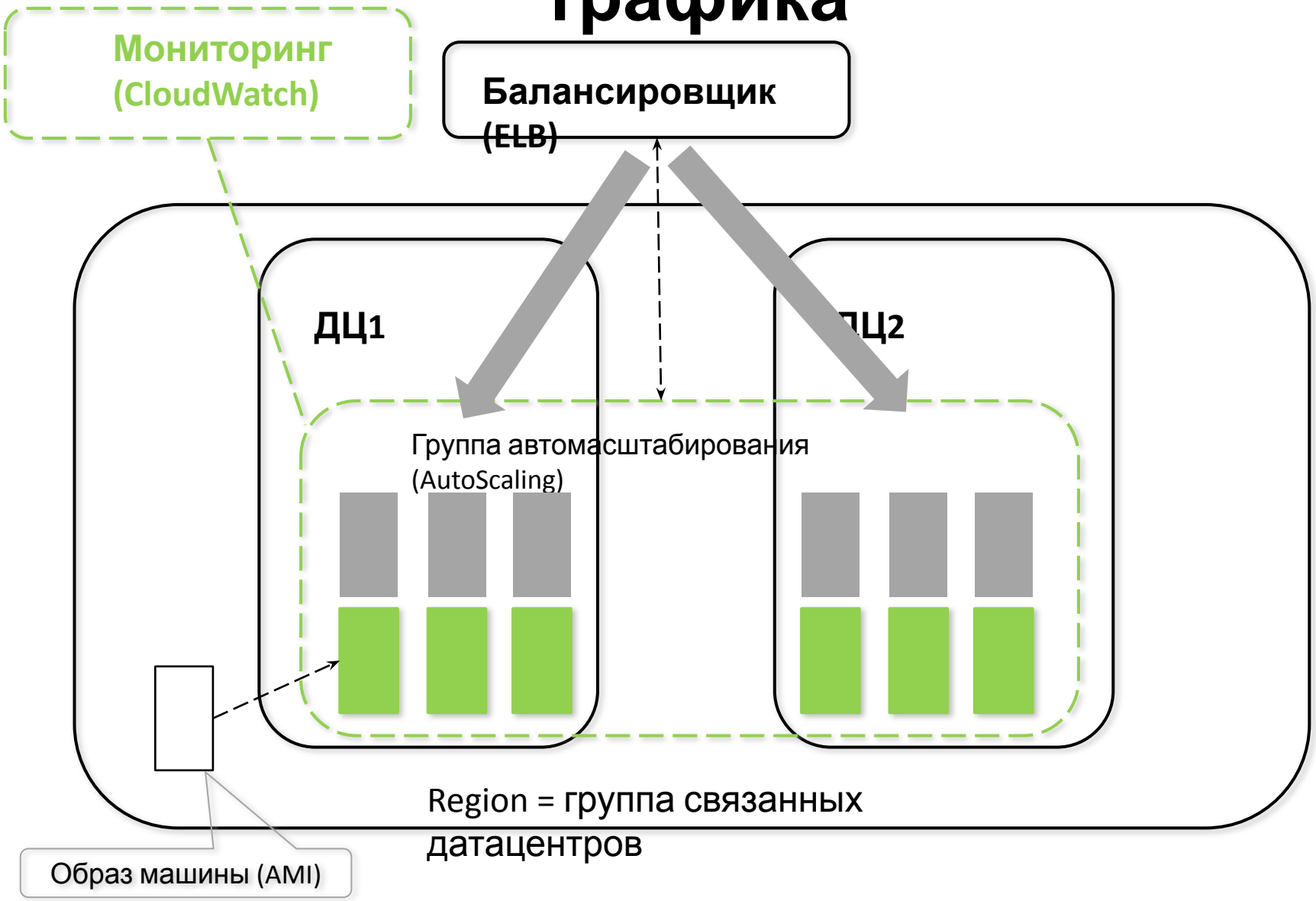
# Балансировка/Переключение трафика



# Балансировка/Переключение трафика



# Балансировка/Переключение трафика



# Балансировка/Переключение трафика

- Можно гибко управлять тремя службами: ELB, AutoScaling, CloudWatch – подстраивая функционал под свои задачи
- Автоматическое увеличение числа машин при росте нагрузки
- Автоматическая замена вышедших из строя машин
- Переключение трафика в другой ДЦ при аварии сервиса или для проведения регламентных работ

# Балансировка/Переключение трафика

AWS SDK for PHP:

AmazonELB:: **set_load_balancer_listener_ssl_certificate** ( $load_balancer_name, $load_balancer_port, $ssl_certificate_id, $opt )

AmazonELB:: **configure_health_check** ( $load_balancer_name, $health_check, $opt )

AmazonELB::**enable_availability_zones_for_load_balancer** ( $load_balancer_name, $availability_zones, $opt )

AmazonELB::**register_instances_with_load_balancer** ( $load_balancer_name, $instances, $opt )

AmazonAS:: **update_auto_scaling_group** ( $auto_scaling_group_name, $opt )

AmazonAS: **set_desired_capacity** ( $auto_scaling_group_name, $desired_capacity, $opt )

и другие.

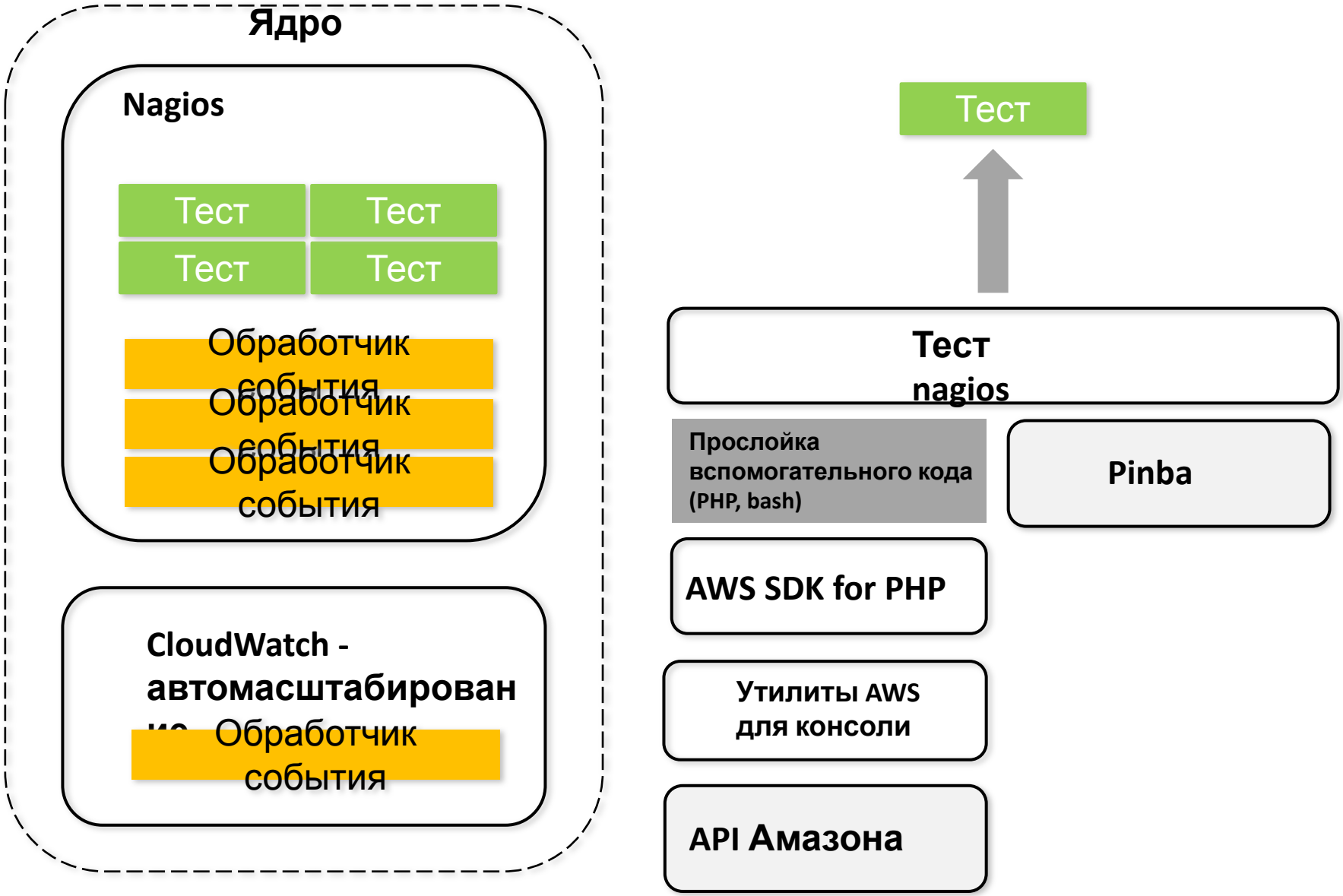
# Автоматическое масштабирование

- В CloudWatch создаем Alarm, который при среднем CPU>20% вызовет действие по добавлению в AutoScaling Group серверов, например, 2 машин (а также вышлет нам письмо)
- Создаем «обратный» Alarm, который при уменьшении нагрузки на CPU<10% будет убирать по 1 машине

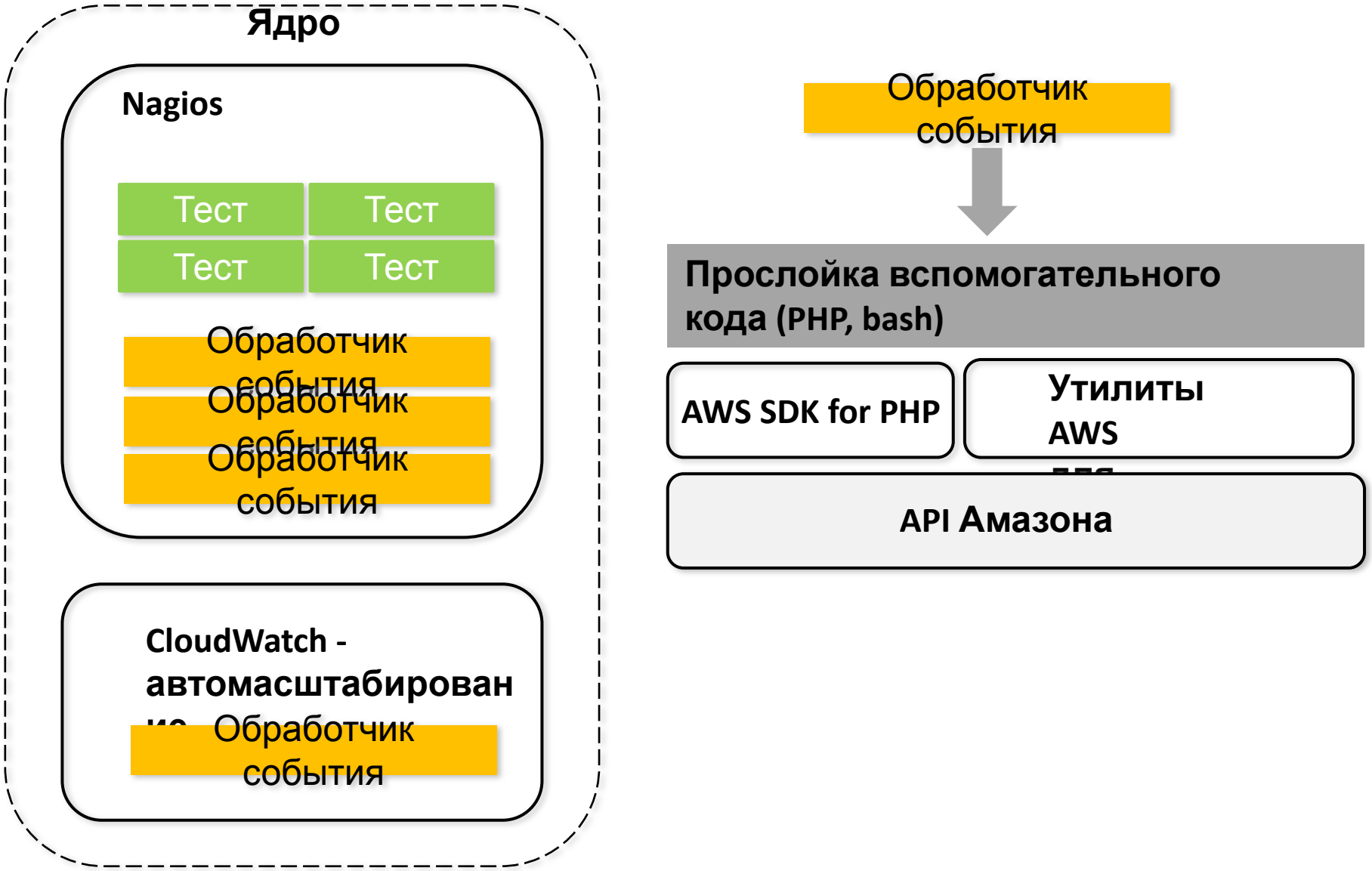
# Система управления

- В CloudWatch недостаточно возможностей, но используем его максимально
- AWS SDK for PHP и вообще работа с API амазона не всегда прямолинейна – нужна прослойка
- Для основного мониторинга и активной обратной связи используем Nagios и его обработчики событий
- Для аналитики в основном используем Munin, часть данных берем из CloudWatch

# Система управления - тест



# Система управления – обработчик события



# Что мы тестируем

Стандартные тесты nagios:

- Пинг, LA, место на дисках
- Использование swap, число процессов
- Мониторинг raid (у нас 10 рейды из EBS-дисков)
- Безопасность – наличие обновлений софта

Для общей картины - набор стандартных плагинов к Munin и несколько десятков графиков по каждой машине

# Что мы тестируем

Базовые тесты MySQL:

- Состояние репликации – отсутствие ошибок, величина отставания slaves
- Buffer Pool hitrate
- Число активных потоков
- Число «долгих» потоков
- Частота создания на диске временных таблиц

и другие

# Что мы тестируем

Тесты времени выполнения страниц и использования памяти от pinba:

- Пиковое время выполнения страницы вирт. хоста
- Пиковое использование памяти вирт. хоста
- Среднее время выполнения страниц вирт. хоста с префиксом (по разделам)
- Среднее использование памяти вирт. хостом
- Среднее время выполнения агентов (после fastcgi_finish_request)

Дополнительно выводим гистограмму распределения хитов по ступенькам времени (по pinba и по суточным логам)

# Что мы тестируем

Тесты состояния амазона и превышения лимитов:

- Проверяем превышения ключевых лимитов амазона
- Число живых машин за балансировщиками
- Планируем проверять показатели нагрузки от CloudWatch – их, однако, очень мало
- Пинг на машину мониторинга nagios в другом регионе амазона и обратно (мониторинг мониторинга ☺)

# Что мы тестируем

Тесты выполнения бизнес-операций и обработчиков событий:

- Скрипты бэкапов и обработчики обновляют лог-файл (проверка времени модификации)
- Скрипты бэкапов и обработчики в обязательном порядке имеют лог ошибок (2> или error_log) – он должен быть пустым

Лог ошибок очень помогает при работе с иногда «нестабильным» API амазона:

- Недоступность веб-сервиса
- Недоступность ресурса, хотя уже имеется его ID

# Администрирование через тестирование

Перед добавлением любого сервиса или обработчика в обязательном порядке:

- Добавляется тест на проверку его «живости», иногда на число процессов с данным именем и т. п.
- Добавляется тест на наличие и своевременное обновление лог-файла обработчика/сервиса
- Добавляется тест на нулевой размер лога ошибок

Сейчас в системе: около 1000 тестов, 5 сервисных групп, 4 групп хостов.

Это конечно замедляет, однако позволяет уверенно двигаться вперед, менять конфигурацию

# Обработчики

Обработчики стараются вернуть систему в рабочее состояние:

- При недоступности memcached(ов) делается временное переключение трафика в другой ДЦ
- При крахе машины MySQL или процесса, репликации – временное переключение трафика в другой ДЦ
- Проводится простая «реанимация» (рестарт memcached) и т.п.
- Если все сервисы поднялись – трафик возвращается обратно

Обработчики для nagios написаны на bash, PHP/MySQL

# Pacemaker/Heartbeat vs Nagios

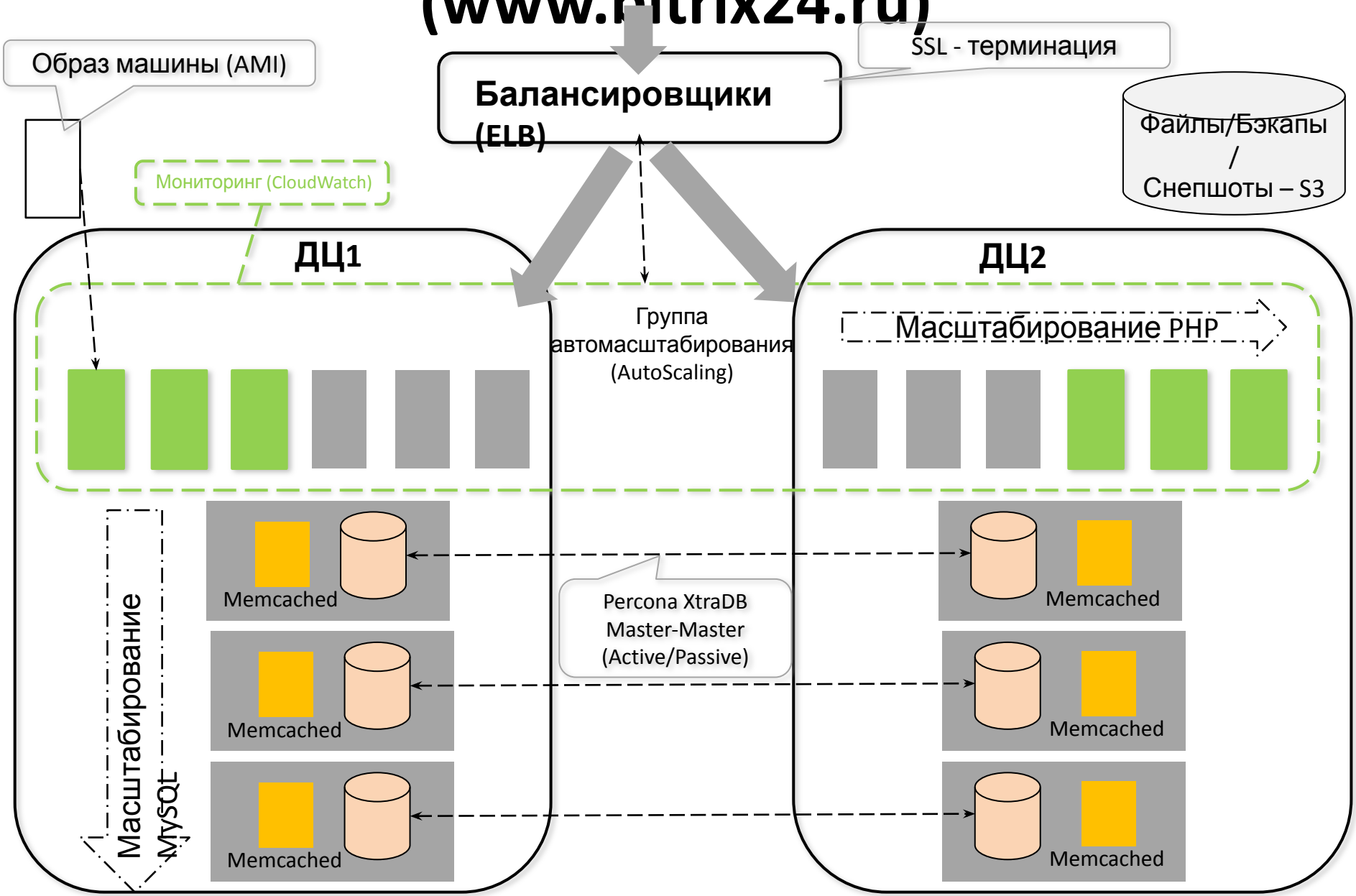
Было несколько безуспешных подходов понять стройность и красоту – Pacemaker/Heartbeat. Документация – ужасна.

Пока нет доверия DRBD.

Пишем свою простую автоматику на базе Nagios/PHP/MySQL и AWS ☺

Используем для кластера AutoScaling, шарды MySQL Master-Master (Active-Passive) и переключение трафика на балансировщиках (можно также Elastic IP). Картинка – ниже

# Архитектура «Битрикс24» (www.bitrix24.ru)



# Спасибо за внимание

Попробуйте http://www.bitrix24.ru

12 человек/5ГБ - бесплатно

Александр Сербул

1С-Битрикс

serbul@1c-bitrix.ru

@AlexSerbul