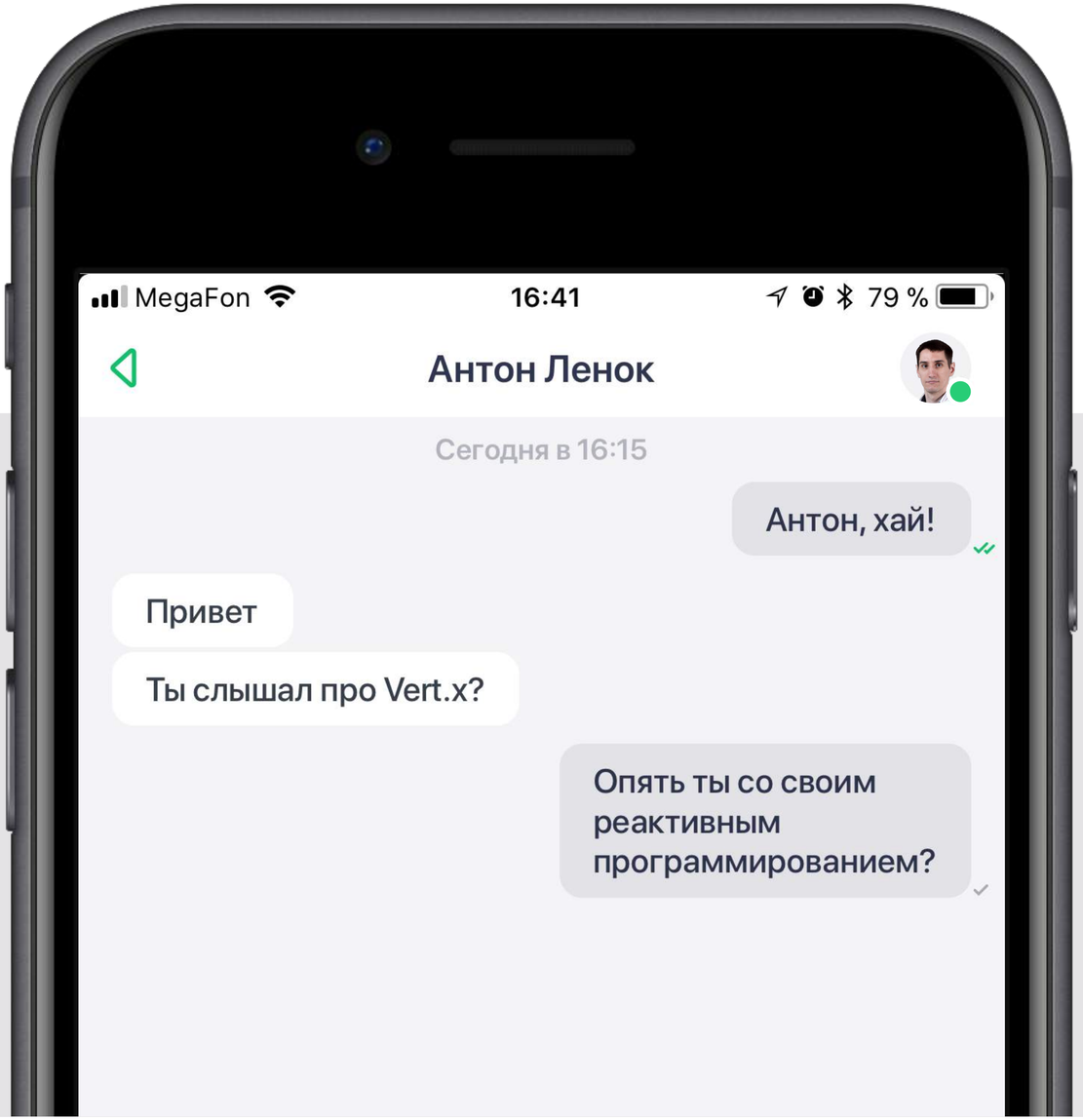
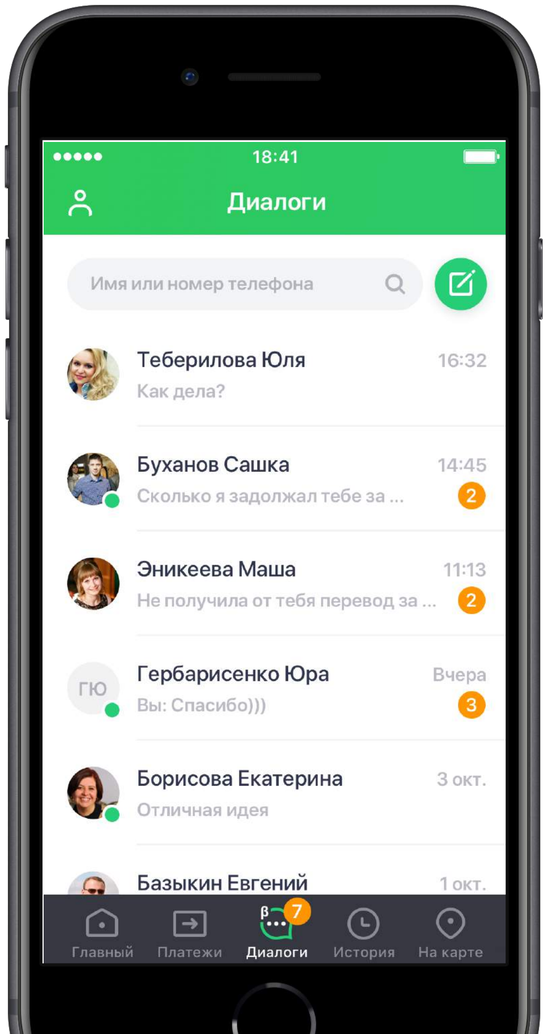
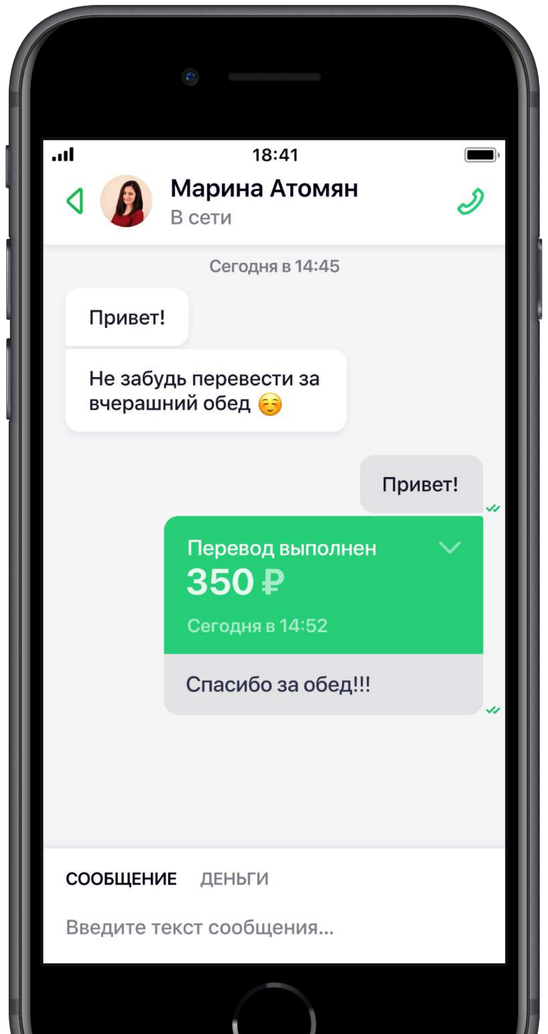
Сбербанк Онлайн

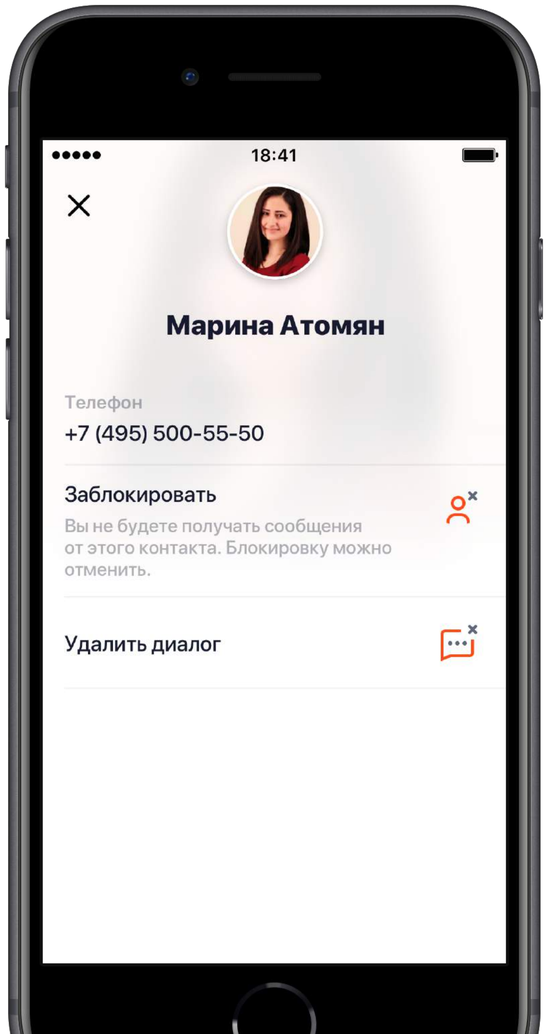
Реактивное программирование на Vert.x

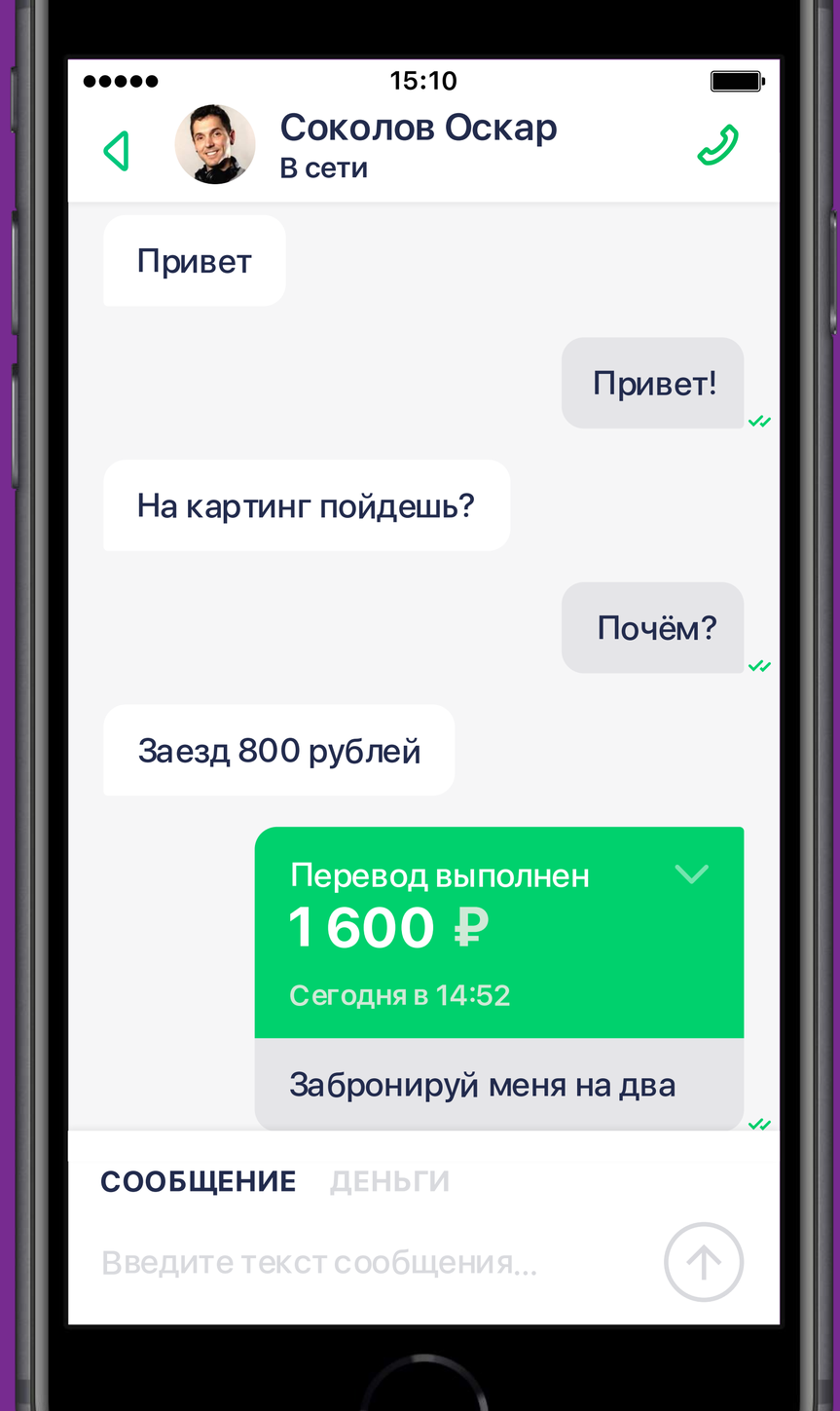


Диалоги в Сбербанк Онлайн








15:10
Соколов Оскар
В сети
Привет
Привет!
На картинг пойдешь?
Почём?
Заезд 800 рублей
Перевод выполнен
1 600 ₽
Сегодня в 14:52
Забронируй меня на два
СООБЩЕНИЕ
ДЕНЬГИ
Введите текст сообщения...

Ленок Антон

Серверная разработка

Проект: Диалоги
в Сбербанк Онлайн

AILenok.SBT@sberbank.ru

URL: https://github.com/dzx912/jpoint-2018-vertx

Что мы сделаем

- Мини чат
- Запуск и масштабирование в Kubernetes
- Бот API
- Подключение БД

Задача

Прием и отправка сообщений

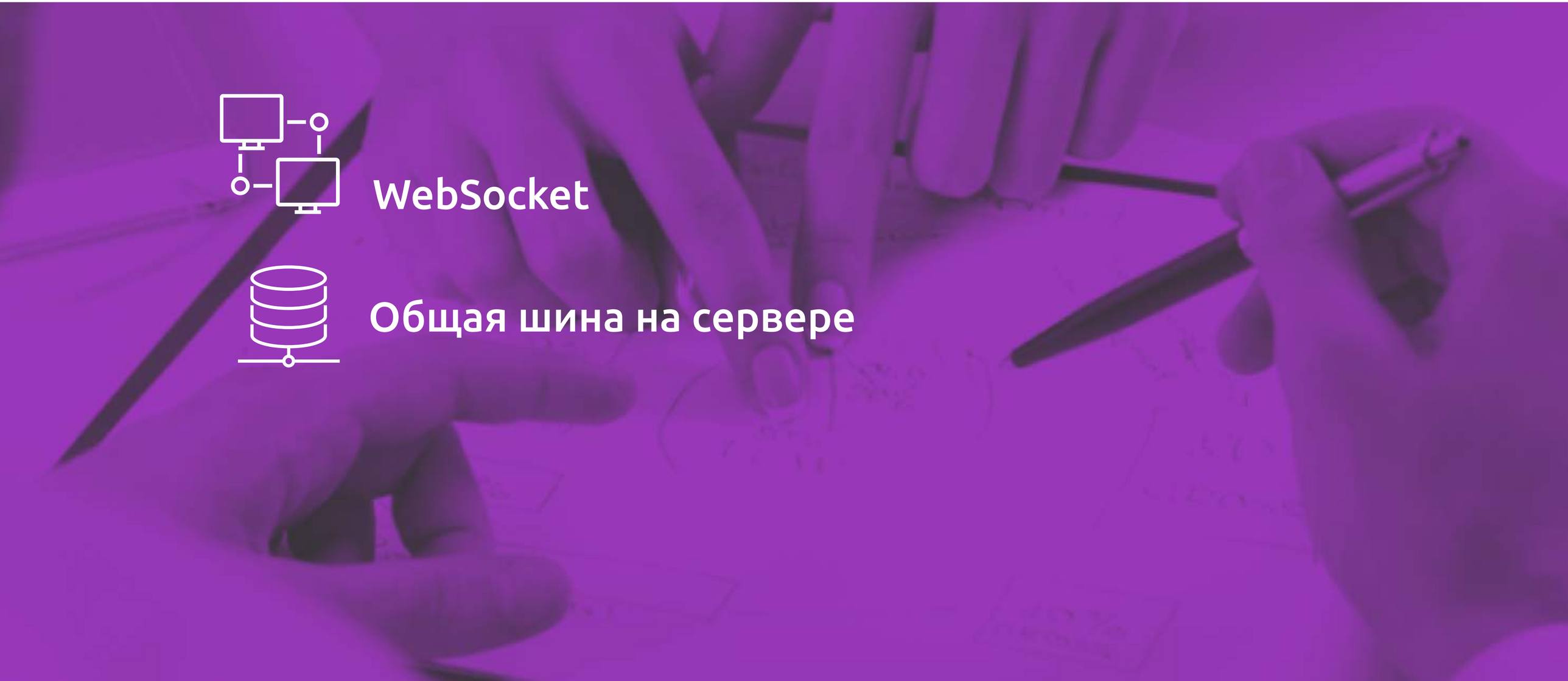
>1 Скорость отправки > приёма сообщения <1 секунды

В начале 10 человек онлайн

Целевое 100 000 человек онлайн

Быстрое добавление функций

Решение



WebSocket

Общая шина на сервере

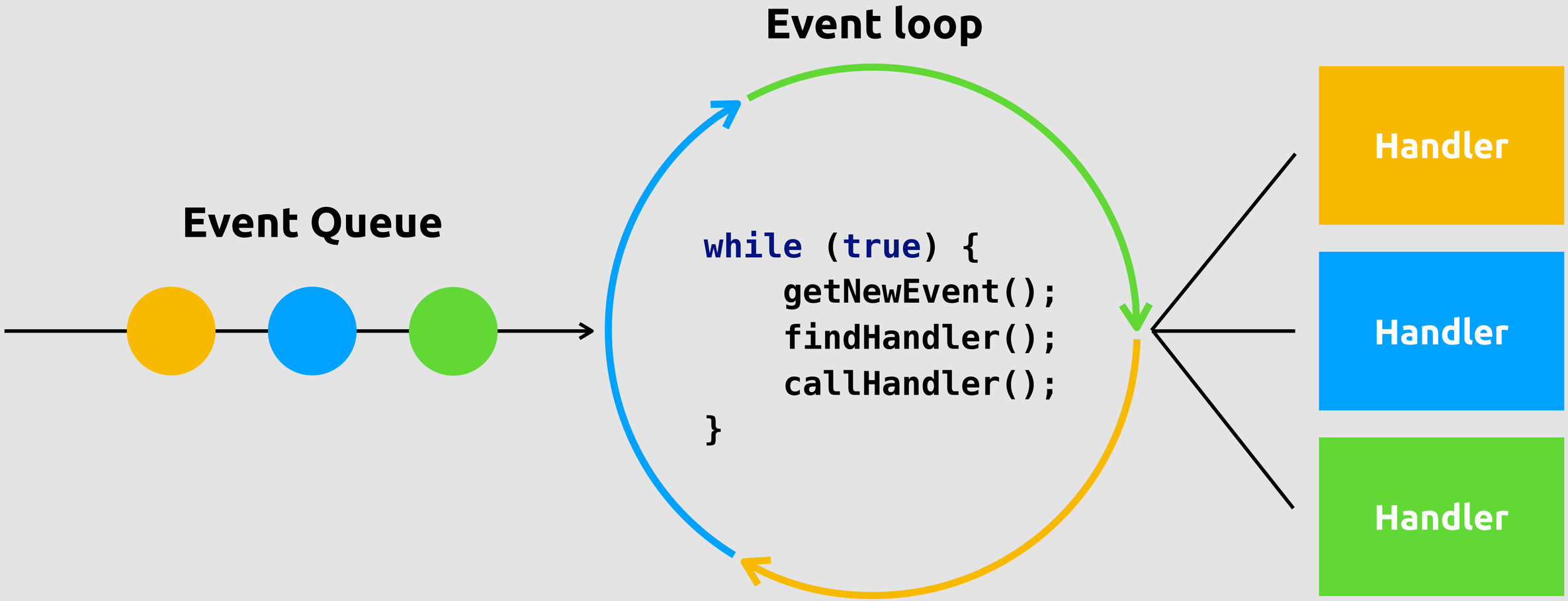
Vert.x описание

- Встроенная общая шина
- Абстракция над Java Concurrency
- Реактивный

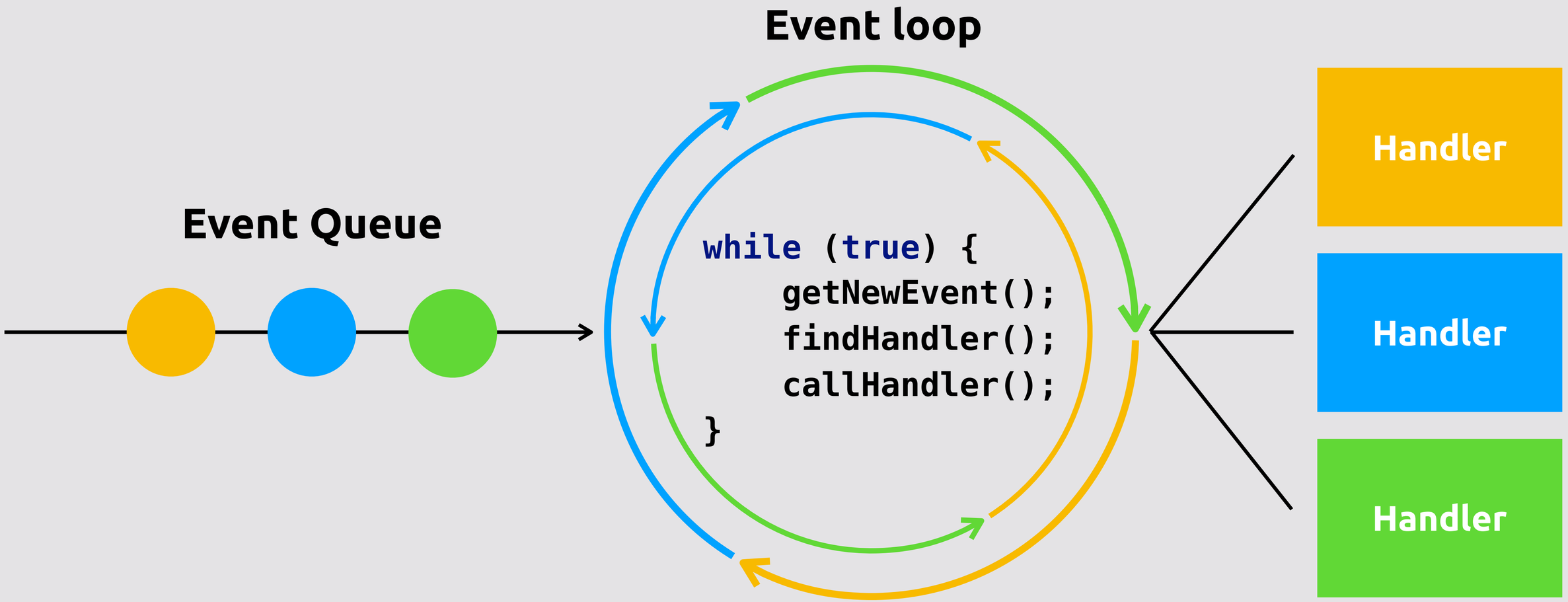
Полиглот для языков JVM

- Java
- JavaScript
- Ruby
- Groovy
- Scala
- Kotlin

Реактивное программирование



Multi-Reactor



Реактивное программирование

Demo

Научились

- Первое приложение

Протокол

```
{
   "text"    : "Hello",
   "address": "nickname"
}
```

Клиент

```
var socket =
new WebSocket(
'ws://HOST/token/ADDRESS'
)
```

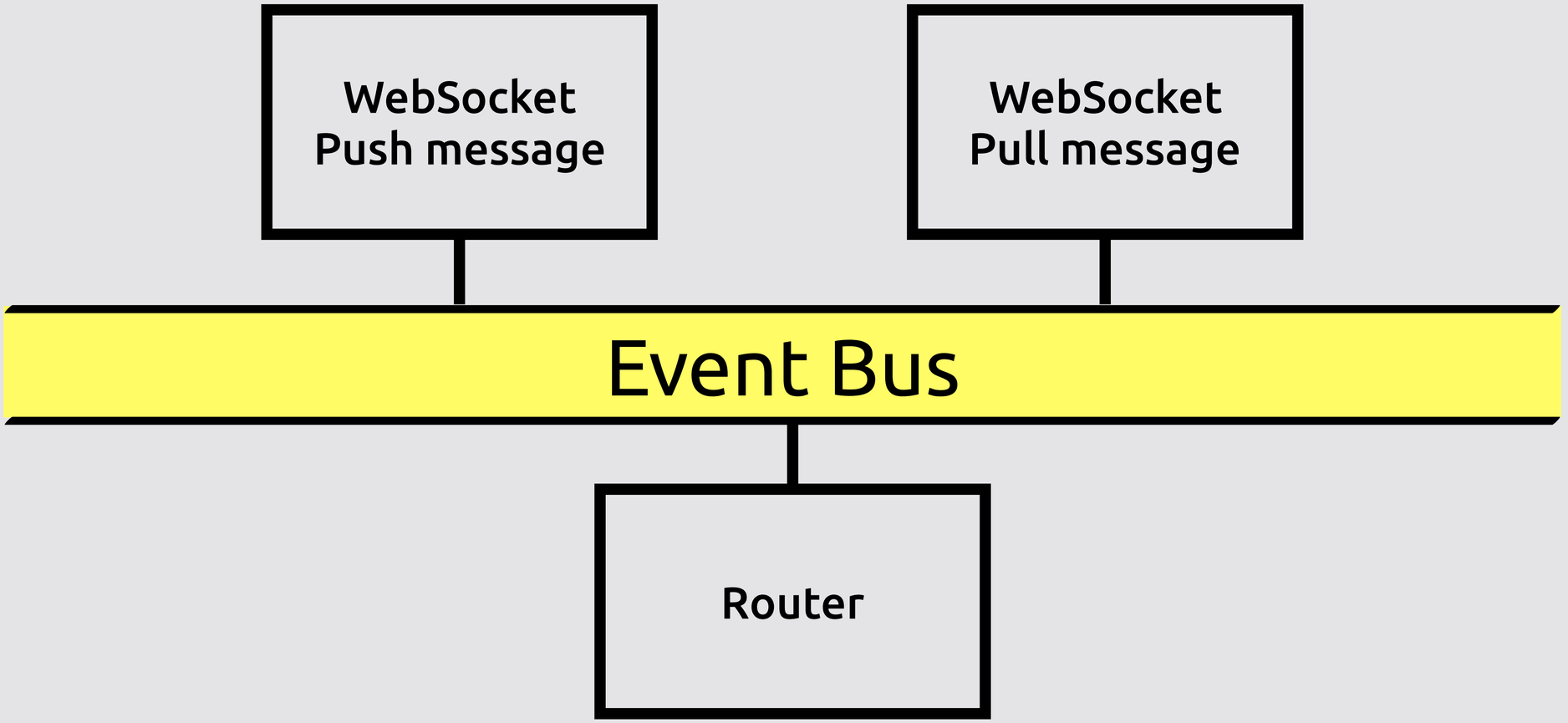
Клиент

```
var socket =
new WebSocket(
'ws://HOST/token/ADDRESS'
)
```

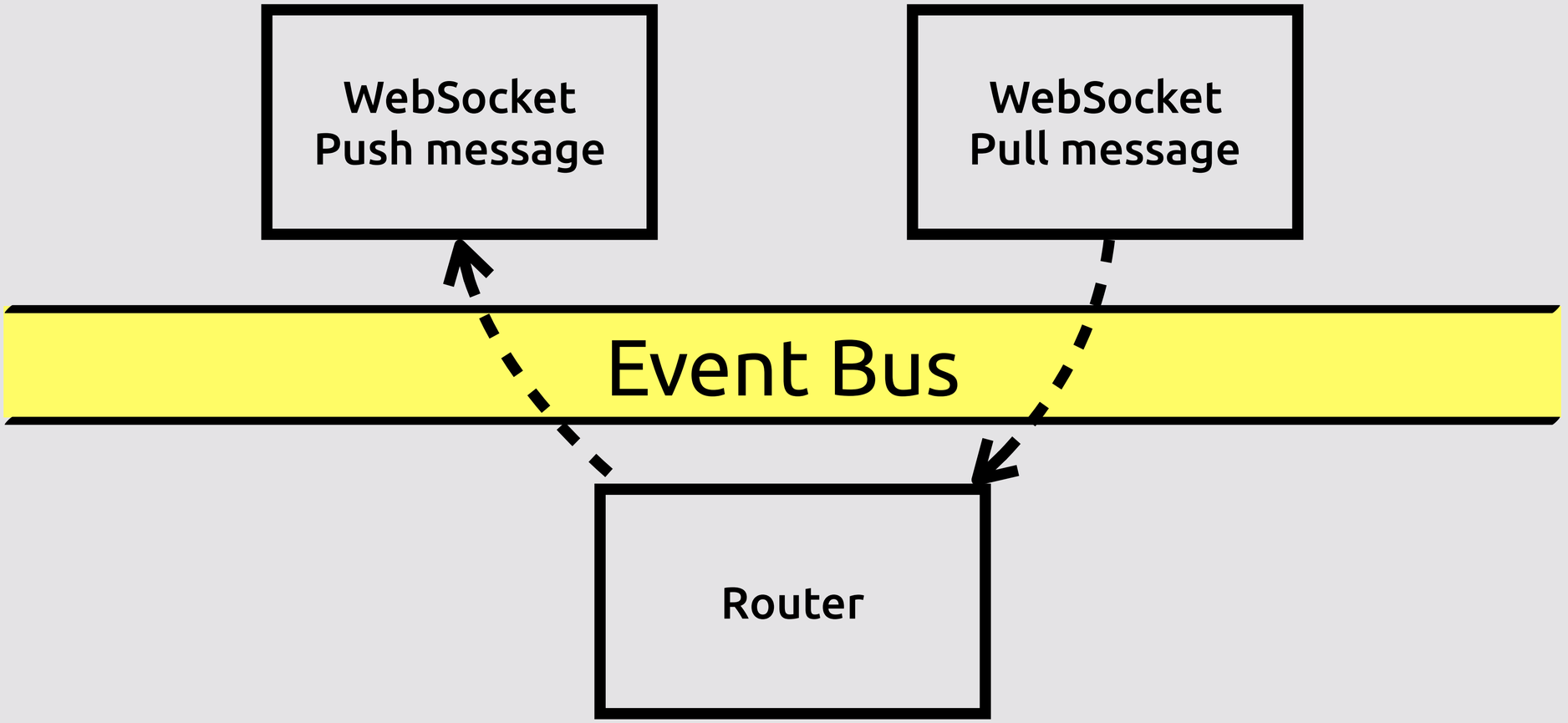
Клиент

```
socket.send(
'{"text"    :"Hello",
  "address":"receiver"}'
)
```

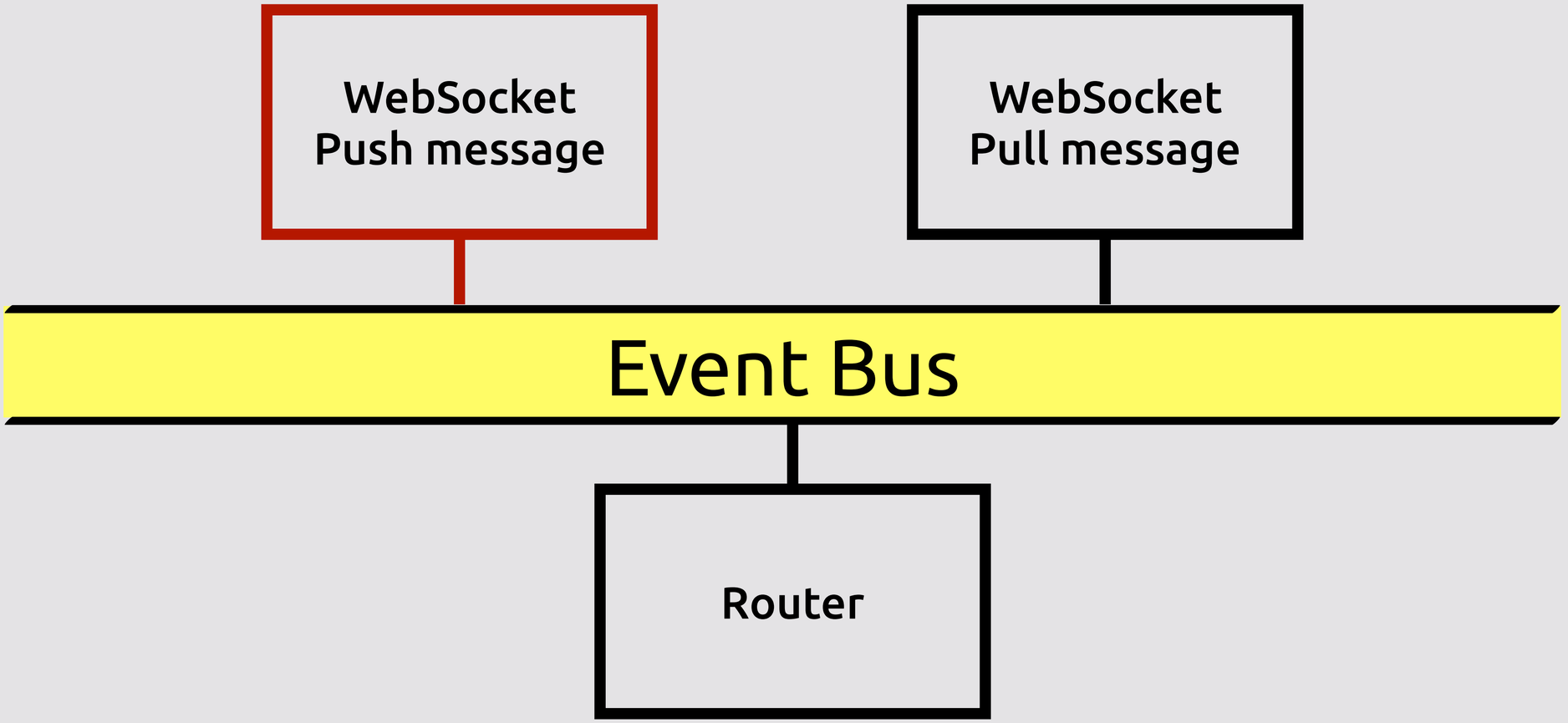
Общая шина v1



Общая шина v1



Общая шина v1



Отправка сообщений


```
public class WsServerVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.createHttpServer()
  .websocketHandler(this::createWebSocketServer)
  .listen(8080);
 }
}
```

Отправка сообщений


```
public class WsServerVerticle extends AbstractVerticle {
 @Override
 public void start() {
   vertx.createHttpServer()
   .websocketHandler(this::createWebSocketServer)
   .listen(8080);
 }
}
```

Отправка сообщений


```
public class WsServerVerticle extends AbstractVerticle {
 @Override
 public void start() {
   vertx.createHttpServer()
   .websocketHandler(this::createWebSocketServer)
   .listen(8080);
 }
}
```

Отправка сообщений


```
public class WsServerVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.createHttpServer()
  .websocketHandler(this::createWebSocketServer)
  .listen(8080);
 }
}
```

Отправка сообщений


```
public class WsServerVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.createHttpServer()
  .websocketHandler(this::createWebSocketServer)
  .listen(8080);
 }
}
void createWebSocketServer(ServerWebSocket wsServer) {
}
```

Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(),
  data -> {
    wsServer.writeFinalTextFrame(data.body());
    data.reply("ok");
 });
}
```

Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(),
  data -> {
    wsServer.writeFinalTextFrame(data.body());
    data.reply("ok");
 });
}
```

Отправка сообщений


```
var socket =
new WebSocket(
'ws://HOST/token/ADDRESS'
)
```

Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(),
  data -> {
    wsServer.writeFinalTextFrame(data.body());
    data.reply("ok");
 });
}
```

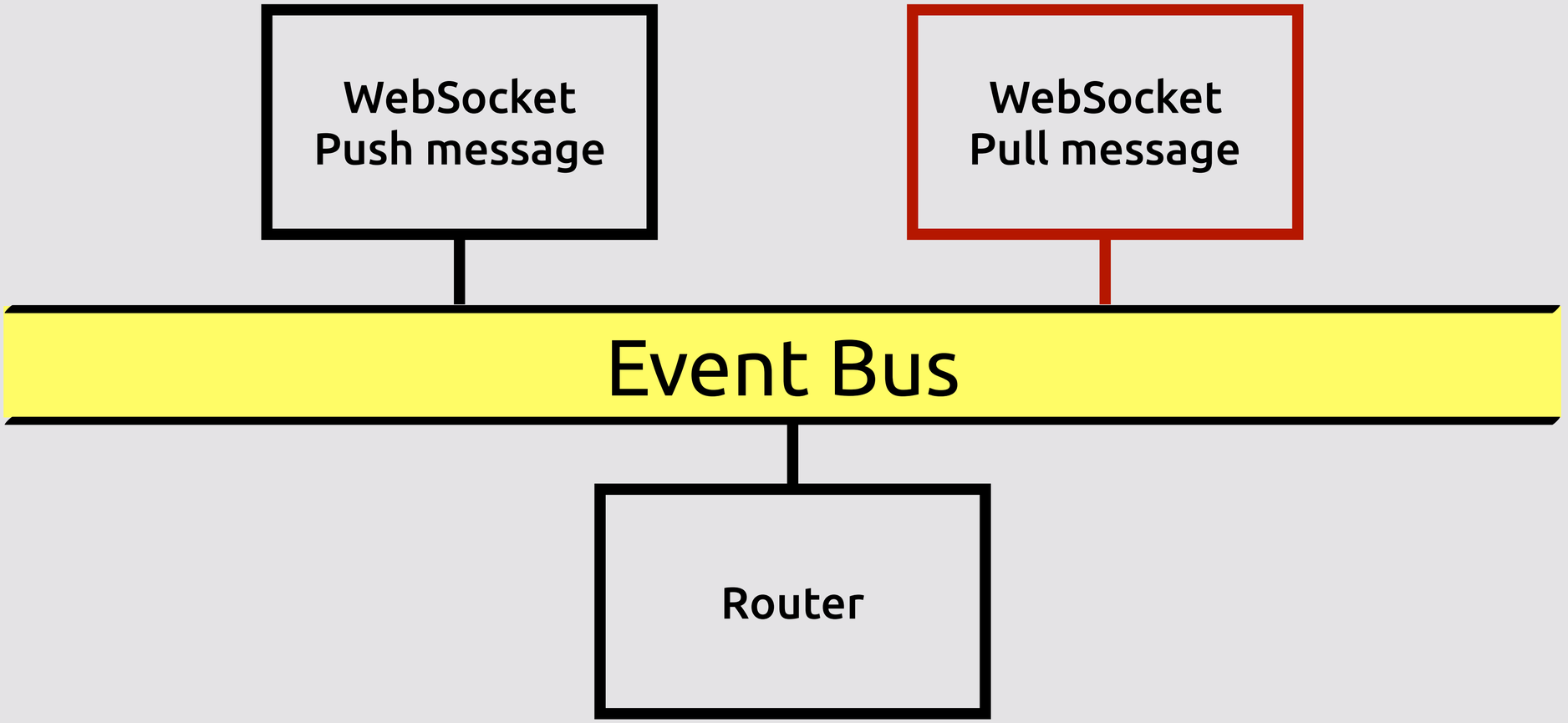
Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(),
  data -> {
    wsServer.writeFinalTextFrame(data.body());
    data.reply("ok");
 });
}
```

Отправка сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(),
  data -> {
    wsServer.writeFinalTextFrame(data.body());
    data.reply("ok");
 });
}
```

Общая шина v1



Прием сообщений


```
public class WsServerVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.createHttpServer()
  .websocketHandler(this::createWebSocketServer)
  .listen(8080);
 }
}
void createWebSocketServer(ServerWebSocket wsServer) {
}
```

Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  vertx.eventBus().send(
  "router", wsFrame.textData());
 });
}
```

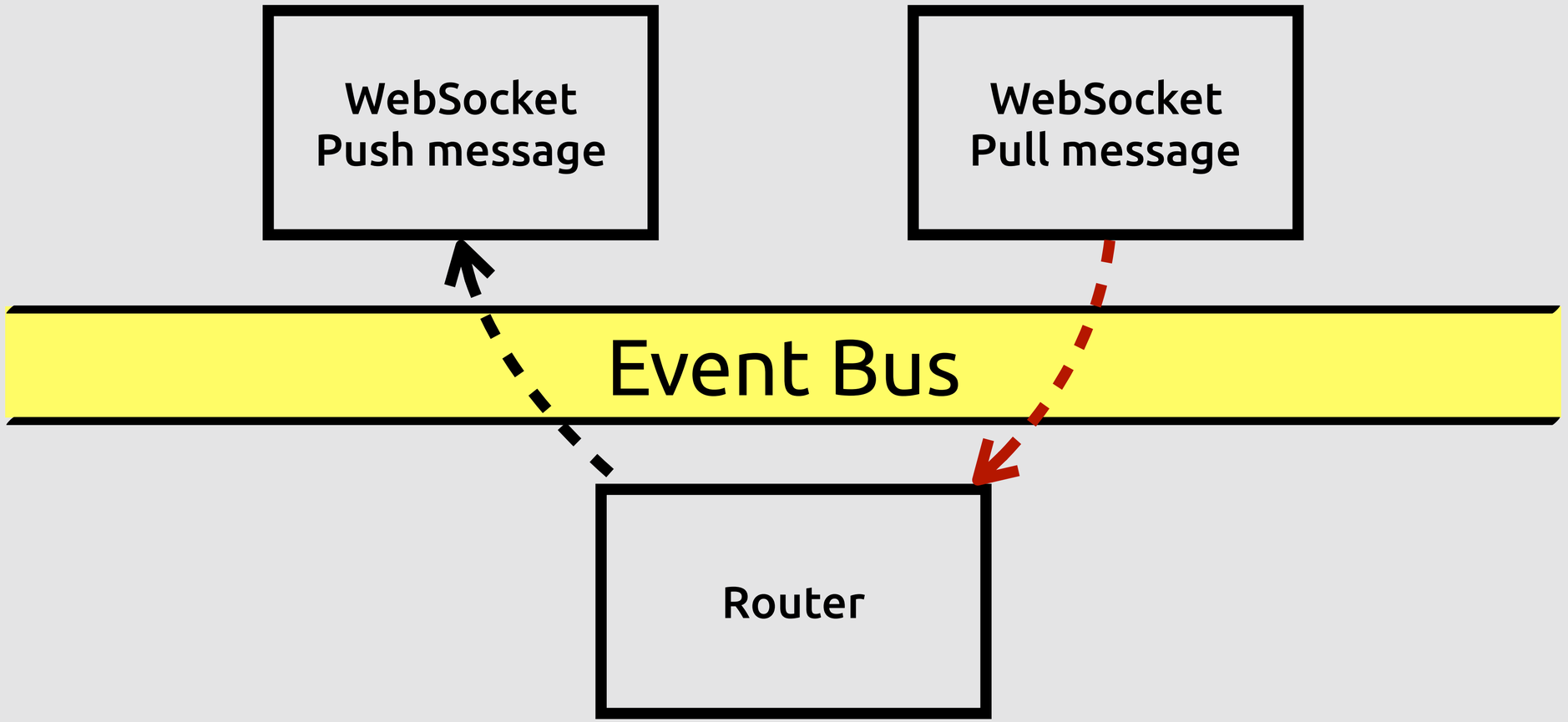
Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  vertx.eventBus().send(
  "router", wsFrame.textData());
 });
}
```

Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  vertx.eventBus().send(
  "router", wsFrame.textData());
 });
}
```

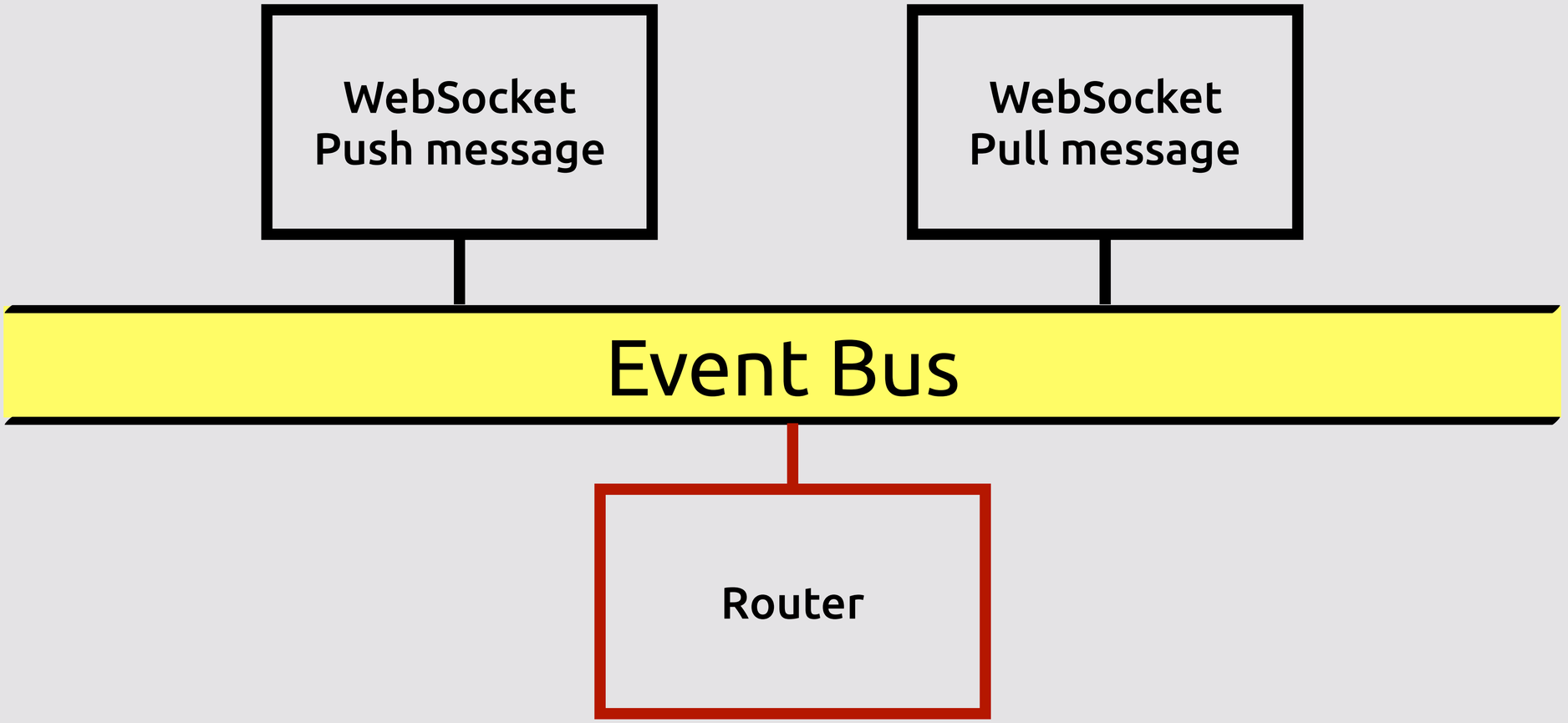
Прием сообщений



Прием сообщений


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  vertx.eventBus().send(
  "router", wsFrame.textData());
 });
}
```

Общая шина v1



Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

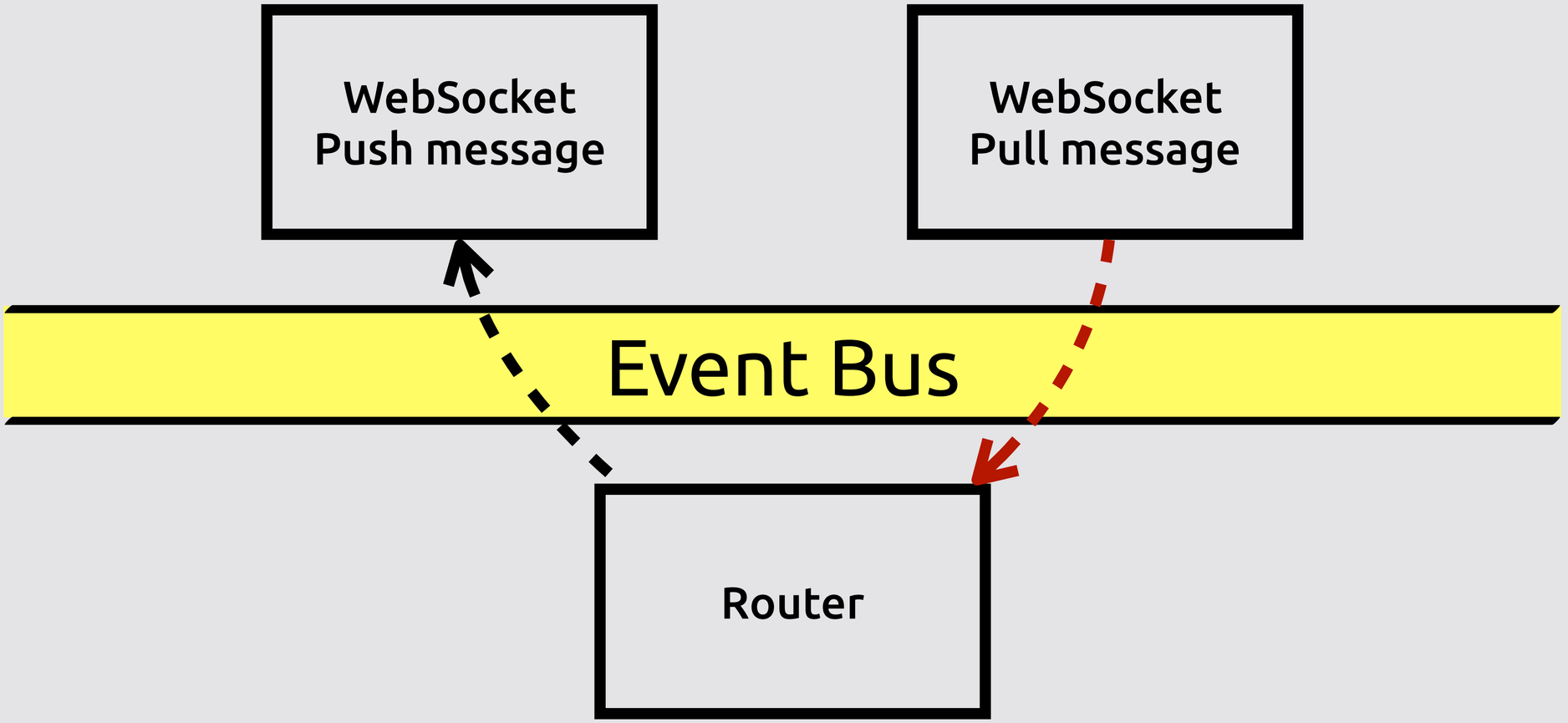
Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  vertx.eventBus().send(
  "router", wsFrame.textData());
 });
}
```

Маршрутизатор



Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор


```
{
   "text"    : "Hello",
   "address": "nickname"
}
```

Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

Маршрутизатор


```
public class RouterVerticle extends AbstractVerticle {
 @Override
 public void start() {
  vertx.eventBus().consumer("router", this::router);
 }

 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());
 }
}
```

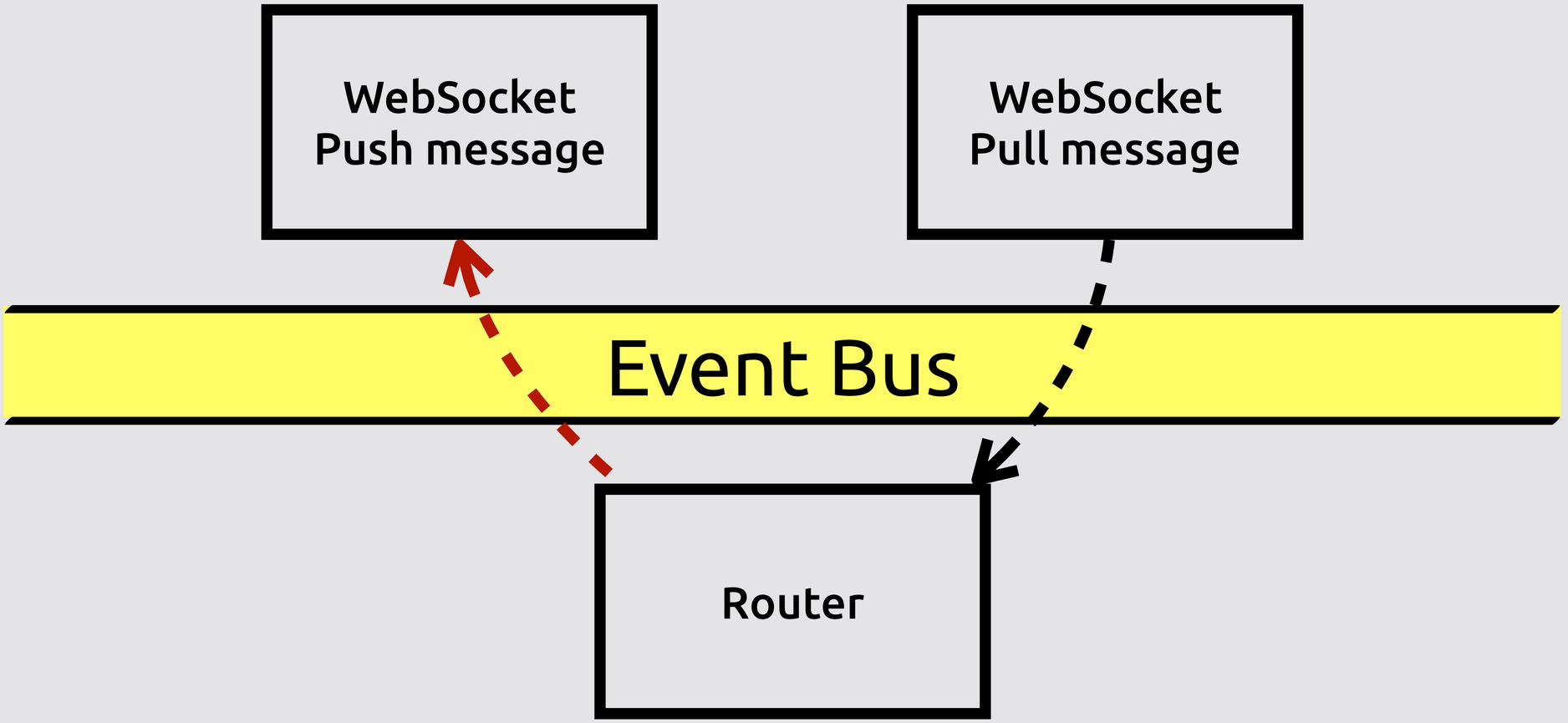
Маршрутизатор


```
{
   "text"    : "Hello",
   "address": "nickname"
}
```

Маршрутизатор


```
void createWebSocketServer(ServerWebSocket wsServer) {
 vertx.eventBus().<String>consumer(wsServer.path(),
  data -> {
    wsServer.writeFinalTextFrame(data.body());
    data.reply("ok");
 });
}
```

Маршрутизатор



Продумывайте маршруты

Kubernetes описание

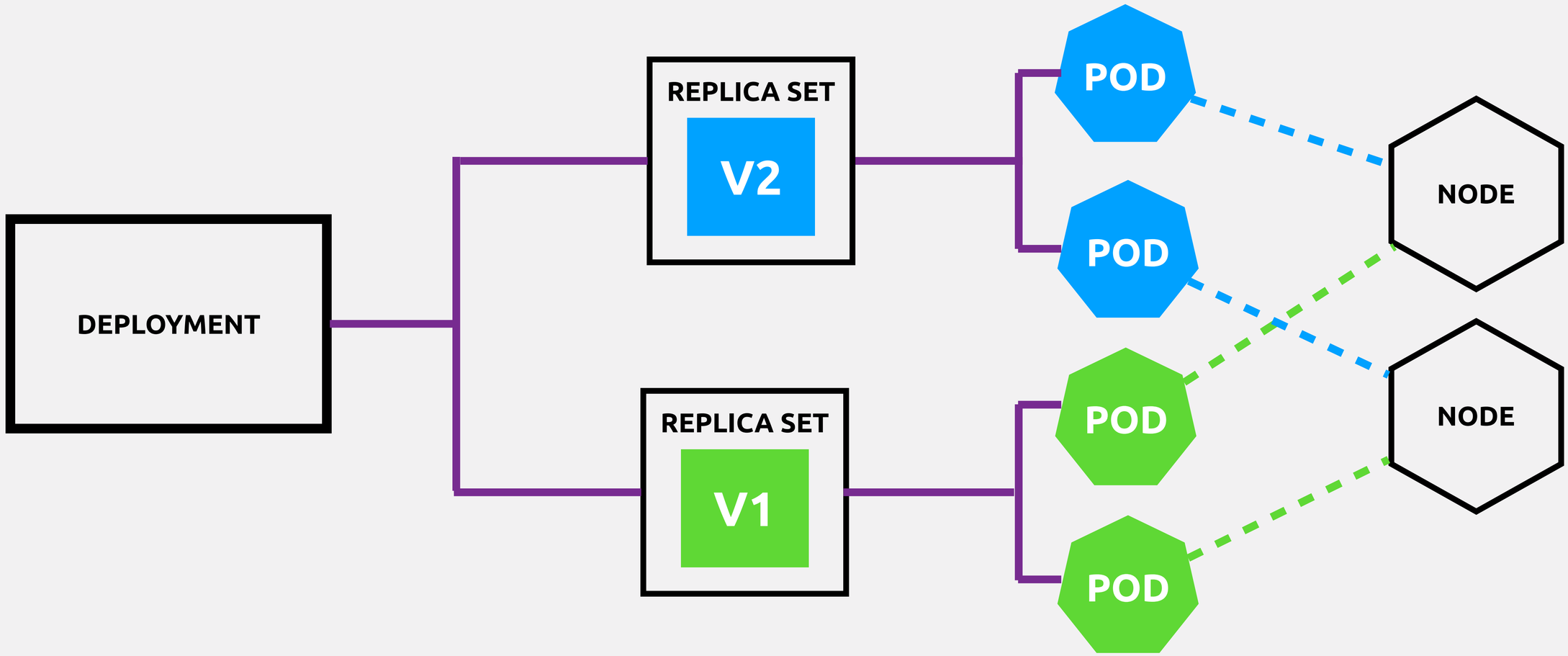
- С греческого — «Рулевой»
- Первоначальный разработчик — Google
- Управляет Docker контейнерами

Kubernetes

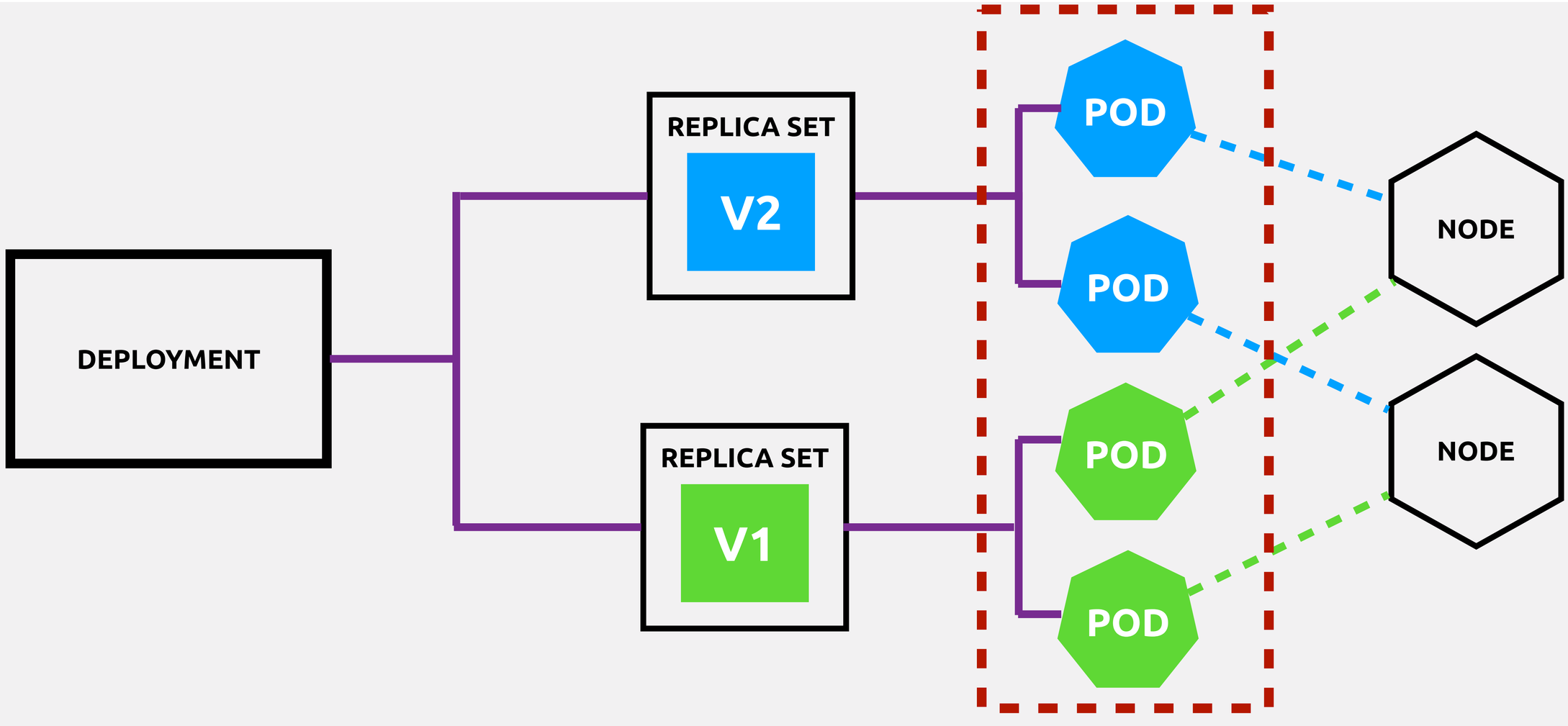
Задача — автоматизировать работу с Docker контейнерами:

- Установка
- Обновление
- Масштабирование
- Возврат к старой версии
- Мониторинг
- Логирование

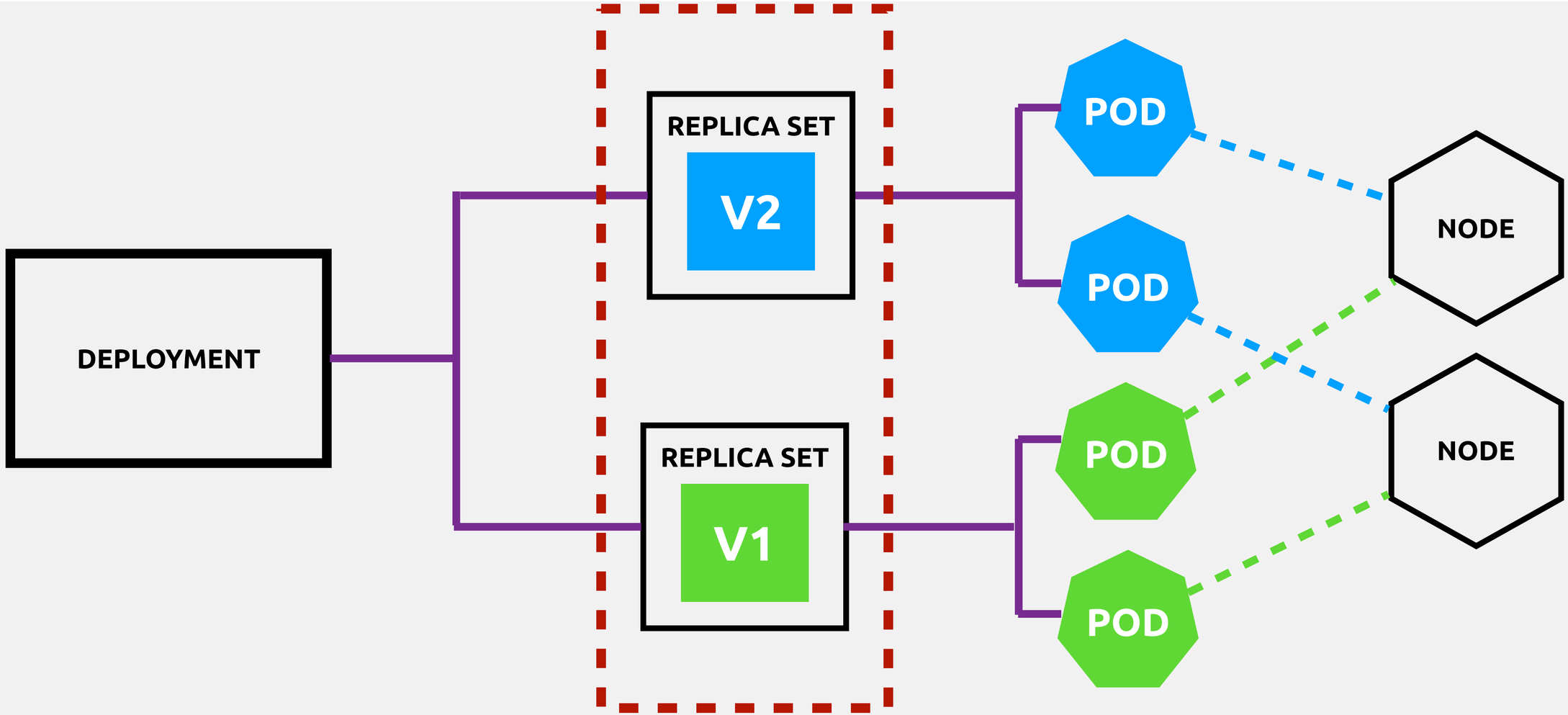
Что такое Deployment



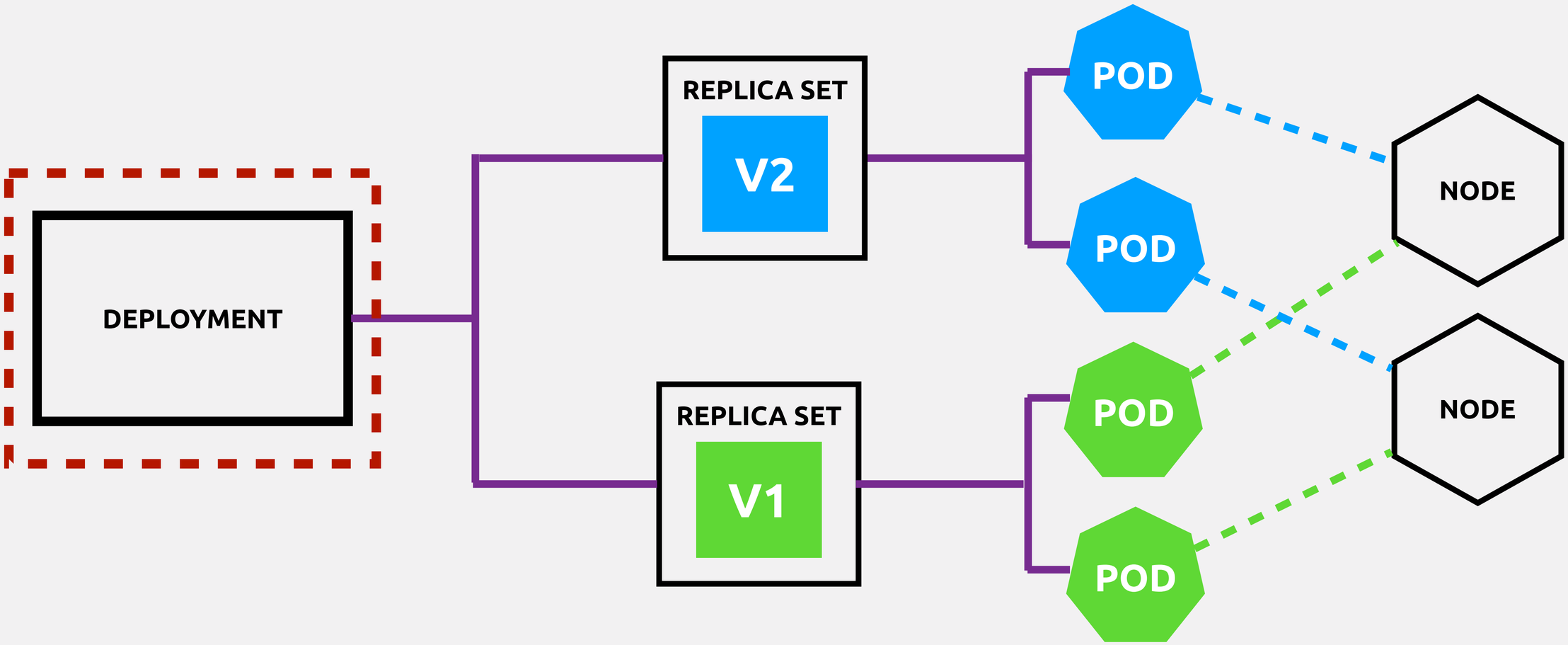
Что такое Deployment



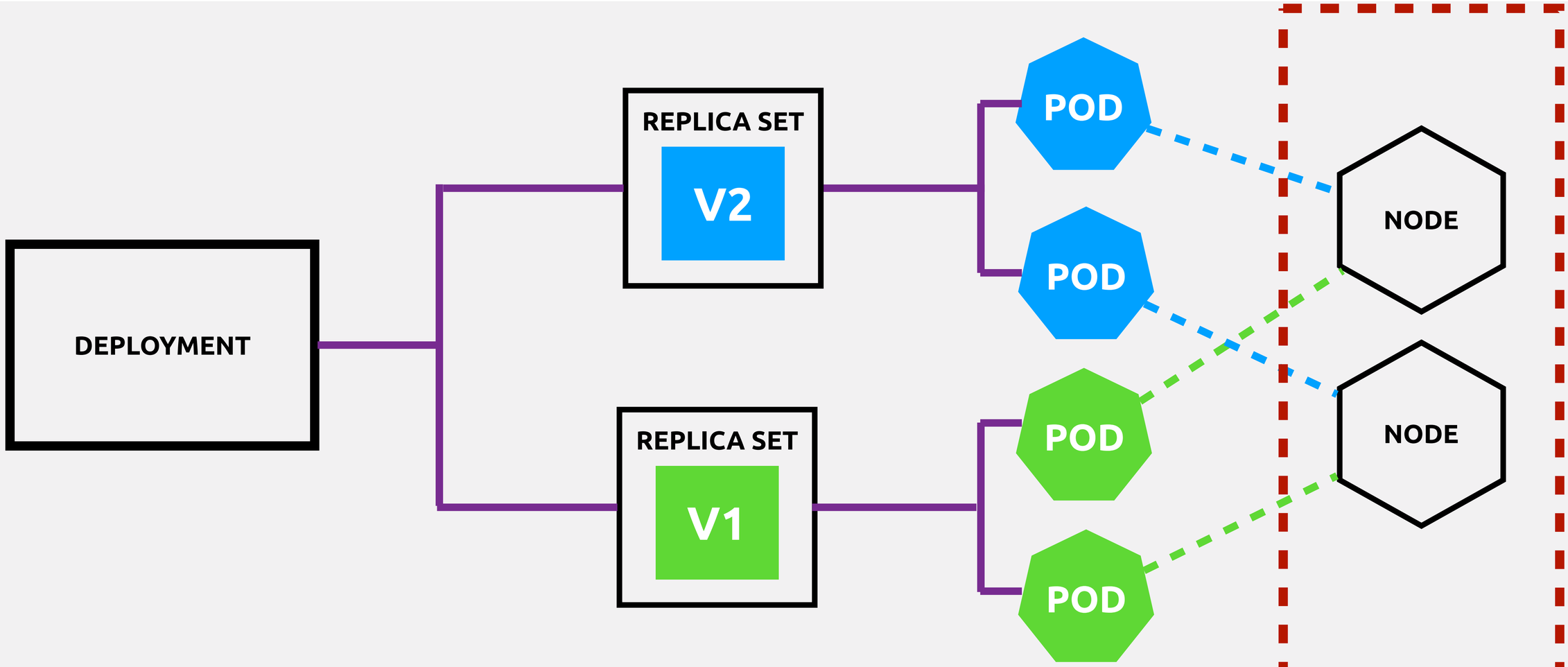
Что такое Deployment



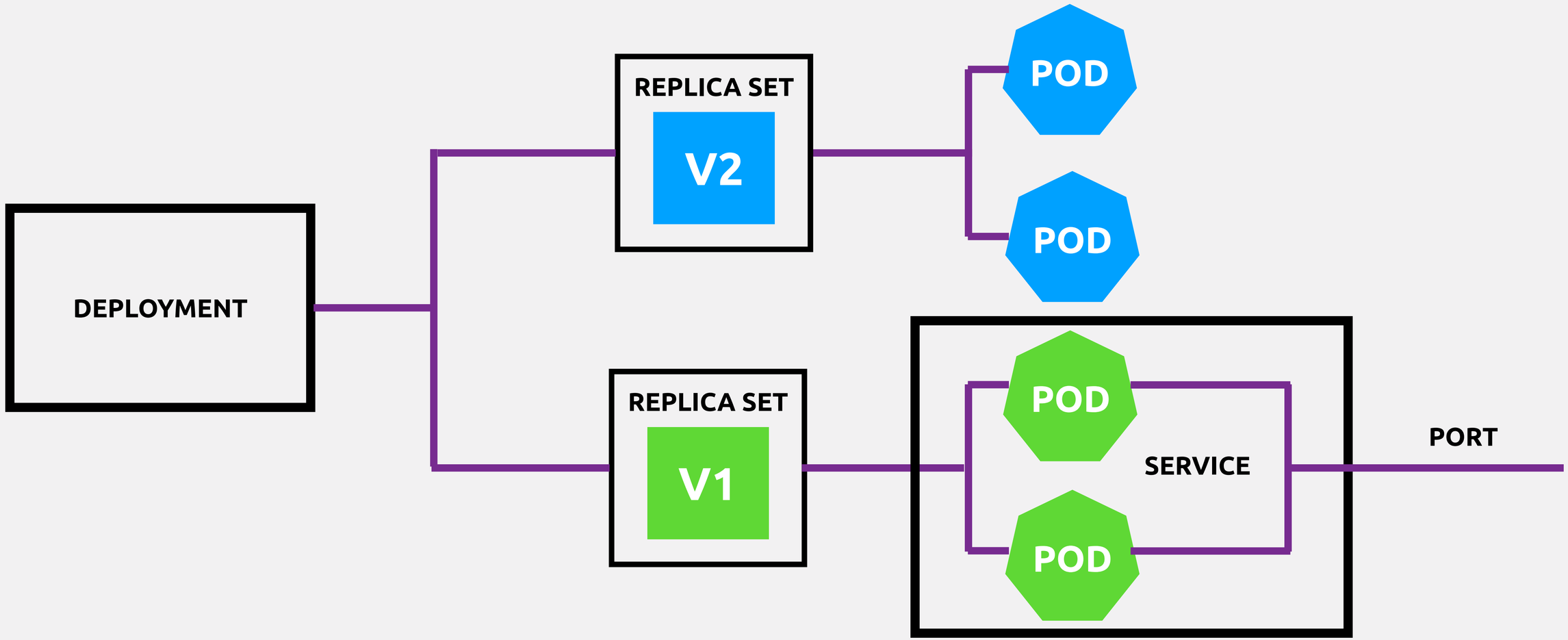
Что такое Deployment



Что такое Deployment



Что такое Service



Deploy


```
public class Starter {

  public static void main(String[] args) {
    Vertx.vertx().deployVerticle(new WsServerVerticle());
    Vertx.vertx().deployVerticle(new RouterVerticle());
  }
}
```

Deploy


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chat-ws-server
spec:
  selector:
    matchLabels:
      run: chat-ws-server
  replicas: 1
  template:
    metadata:
      labels:
        run: chat-ws-server
    spec:
      containers:
      - name: chat-ws-server
        image: dzx912/chat-ws-server:1
        ports:
        - containerPort: 8080
```

Deploy


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chat-ws-server
  replicas: 1
# . . .
    spec:
      containers:
      - name: chat-ws-server
        image: dzx912/chat-ws-server:1
        ports:
        - containerPort: 8080
```

Deploy


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chat-ws-server
  replicas: 1
# . . .
    spec:
      containers:
      - name: chat-ws-server
        image: dzx912/chat-ws-server:1
        ports:
        - containerPort: 8080
```

Deploy


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chat-ws-server
  replicas: 1
# . . .
    spec:
      containers:
      - name: chat-ws-server
        image: dzx912/chat-ws-server:1
        ports:
        - containerPort: 8080
```

Deploy


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chat-ws-server
  replicas: 1
# . . .
    spec:
      containers:
      - name: chat-ws-server
        image: dzx912/chat-ws-server:1
        ports:
        - containerPort: 8080
```

Deploy


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chat-ws-server
  replicas: 1
# . . .
    spec:
      containers:
      - name: chat-ws-server
        image: dzx912/chat-ws-server:1
        ports:
        - containerPort: 8080
```

Deploy


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chat-ws-server
  replicas: 1
# . . .
    spec:
      containers:
      - name: chat-ws-server
        image: dzx912/chat-ws-server:1
        ports:
        - containerPort: 8080
```

Deploy


```
---
apiVersion: v1
kind: Service
metadata:
  name: chat
  labels:
    run: chat
spec:
  selector:
    run: chat-ws-server
  ports:
  - name: websocket
    port: 8080
    nodePort: 30080
  type: LoadBalancer
```

Deploy


```
---
apiVersion: v1
kind: Service
metadata:
  name: chat
  labels:
    run: chat
spec:
  selector:
    run: chat-ws-server
  ports:
  - name: websocket
    port: 8080
    nodePort: 30080
  type: LoadBalancer
```

Deploy


```
---
apiVersion: v1
kind: Service
metadata:
  name: chat
  labels:
    run: chat
spec:
  selector:
    run: chat-ws-server
  ports:
  - name: websocket
    port: 8080
    nodePort: 30080
  type: LoadBalancer
```

Deploy


```
kubectl apply -f chat.yaml
```

Deploy

Demo

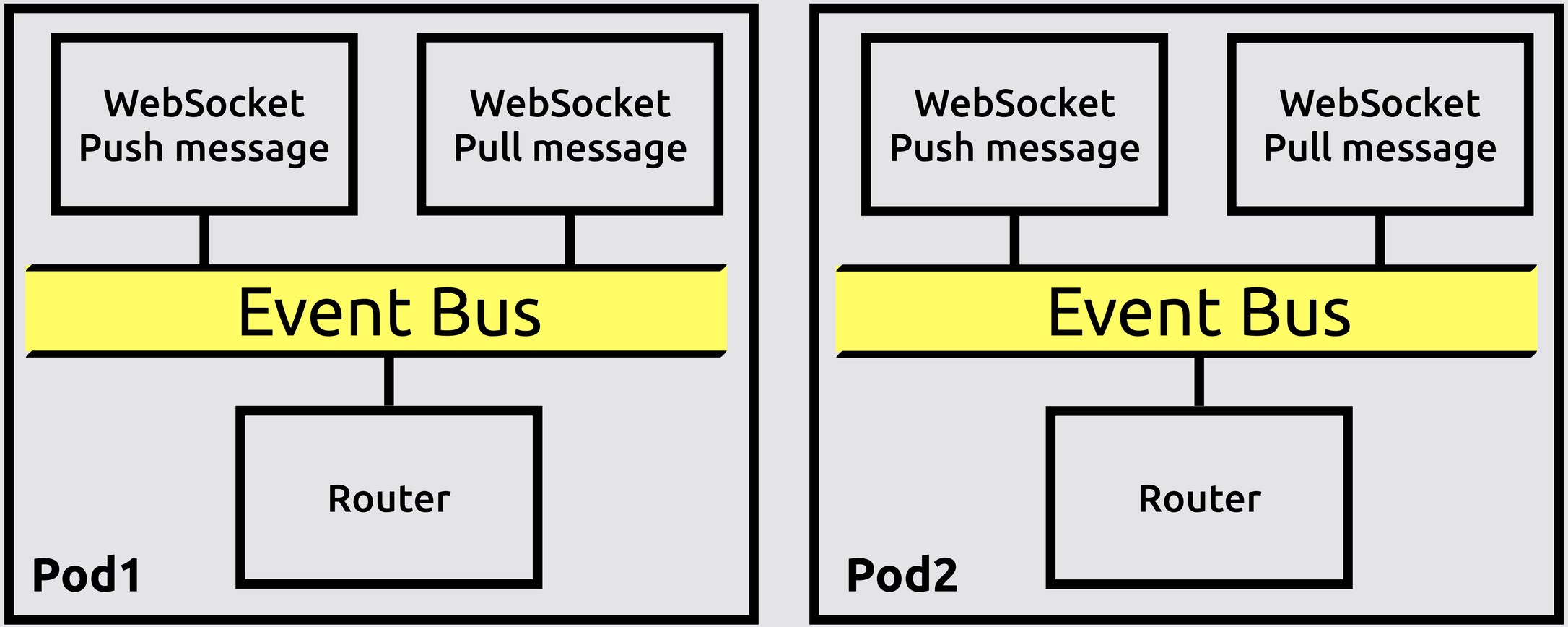
Научились

- Первое приложение
- WebSocket

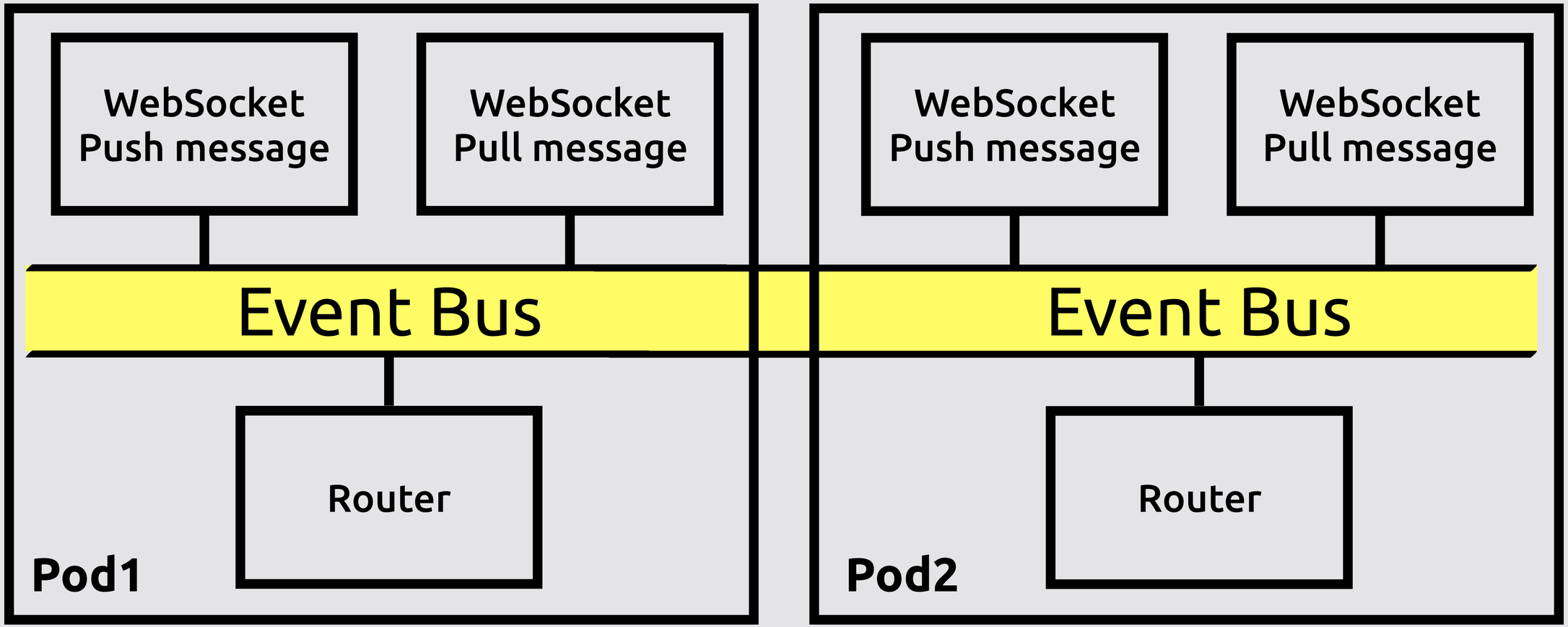
Scale

Demo

Vert.x в Kubernetes



Vert.x в Kubernetes



Cluster Manager в Vert.x

hazelcast

Infinispan

Apache ZooKeeper™

apache Ignite

Cluster в Vert.x


```
IgniteConfiguration getIgniteConfiguration() {
    TcpDiscoverySpi spi = new TcpDiscoverySpi();
    TcpDiscoveryKubernetesIpFinder ipFinder =
            new TcpDiscoveryKubernetesIpFinder();

    ipFinder.setServiceName("chat");

    spi.setIpFinder(ipFinder);
    IgniteConfiguration cfg = new IgniteConfiguration();
    cfg.setDiscoverySpi(spi);
    return cfg;
}
```

Cluster в Vert.x


```
IgniteConfiguration getIgniteConfiguration() {
    TcpDiscoverySpi spi = new TcpDiscoverySpi();
    TcpDiscoveryKubernetesIpFinder ipFinder =
            new TcpDiscoveryKubernetesIpFinder();

    ipFinder.setServiceName("chat");

    spi.setIpFinder(ipFinder);
    IgniteConfiguration cfg = new IgniteConfiguration();
    cfg.setDiscoverySpi(spi);
    return cfg;
}
```

Cluster в Vert.x


```
IgniteConfiguration getIgniteConfiguration() {
    TcpDiscoverySpi spi = new TcpDiscoverySpi();
    TcpDiscoveryKubernetesIpFinder ipFinder =
            new TcpDiscoveryKubernetesIpFinder();

    ipFinder.setServiceName("chat");

    spi.setIpFinder(ipFinder);
    IgniteConfiguration cfg = new IgniteConfiguration();
    cfg.setDiscoverySpi(spi);
    return cfg;
}
```

Cluster в Vert.x


```
---
apiVersion: v1
kind: Service
metadata:
  name: chat
. . . .
```

Cluster в Vert.x


```
IgniteConfiguration getIgniteConfiguration() {
    TcpDiscoverySpi spi = new TcpDiscoverySpi();
    TcpDiscoveryKubernetesIpFinder ipFinder =
            new TcpDiscoveryKubernetesIpFinder();

    ipFinder.setServiceName("chat");

    spi.setIpFinder(ipFinder);
    IgniteConfiguration cfg = new IgniteConfiguration();
    cfg.setDiscoverySpi(spi);
    return cfg;
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x


```
void setupCluster() {
    ClusterManager clusterManager =
        new IgniteClusterManager(getIgniteConfiguration());

    VertxOptions options = new VertxOptions()
        .setClustered(true)
        .setClusterManager(clusterManager)
        .setClusterHost(getMyIp());

    Vertx.clusteredVertx(options, res -> {
        if (res.succeeded()) {
            Vertx vertx = res.result();
            deploy(vertx);
        }
    });
}
```

Cluster в Vert.x

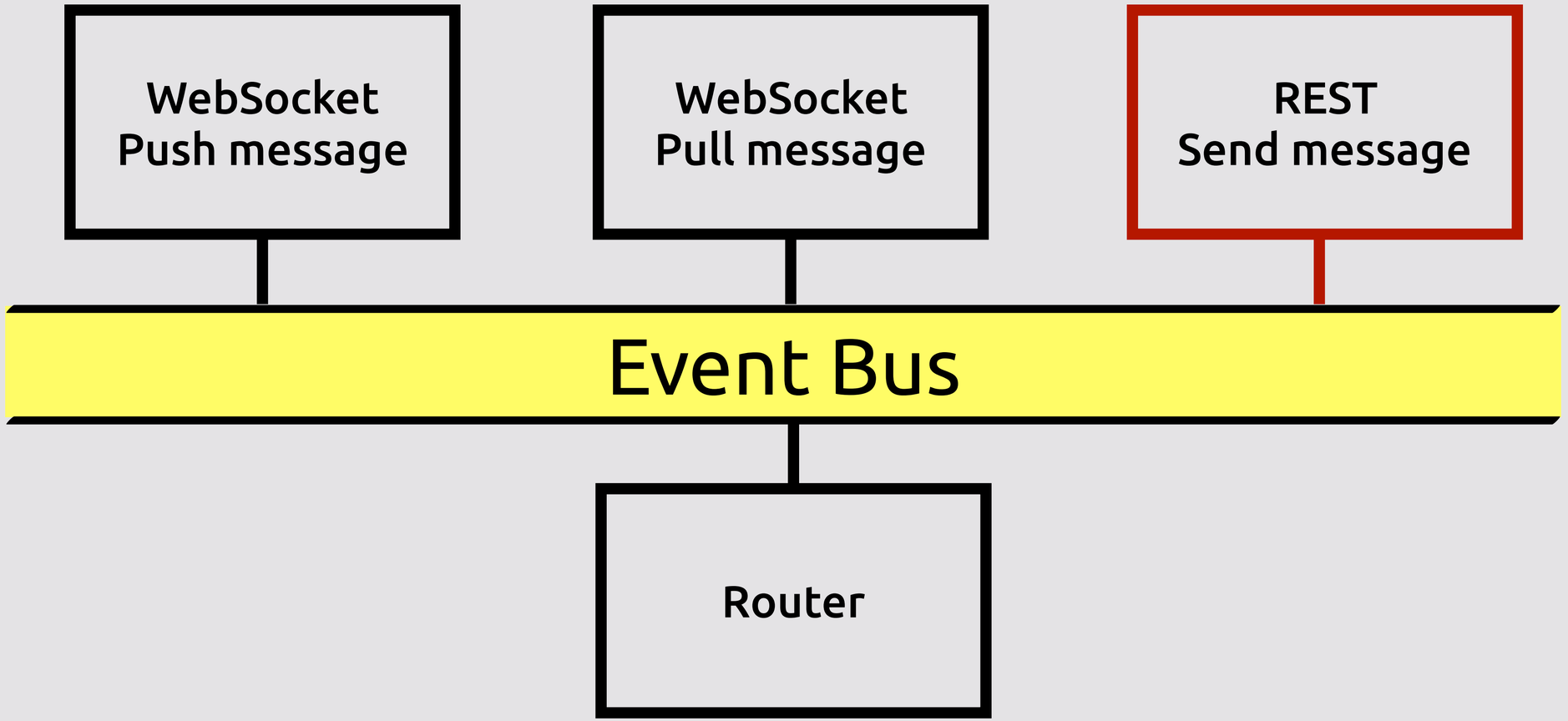

```
void deploy(Vertx vertx) {
    vertx.deployVerticle(new WsServerVerticle());
}

void deploy(Vertx vertx) {
    vertx.deployVerticle(new RouterVerticle());
}
```

Demo

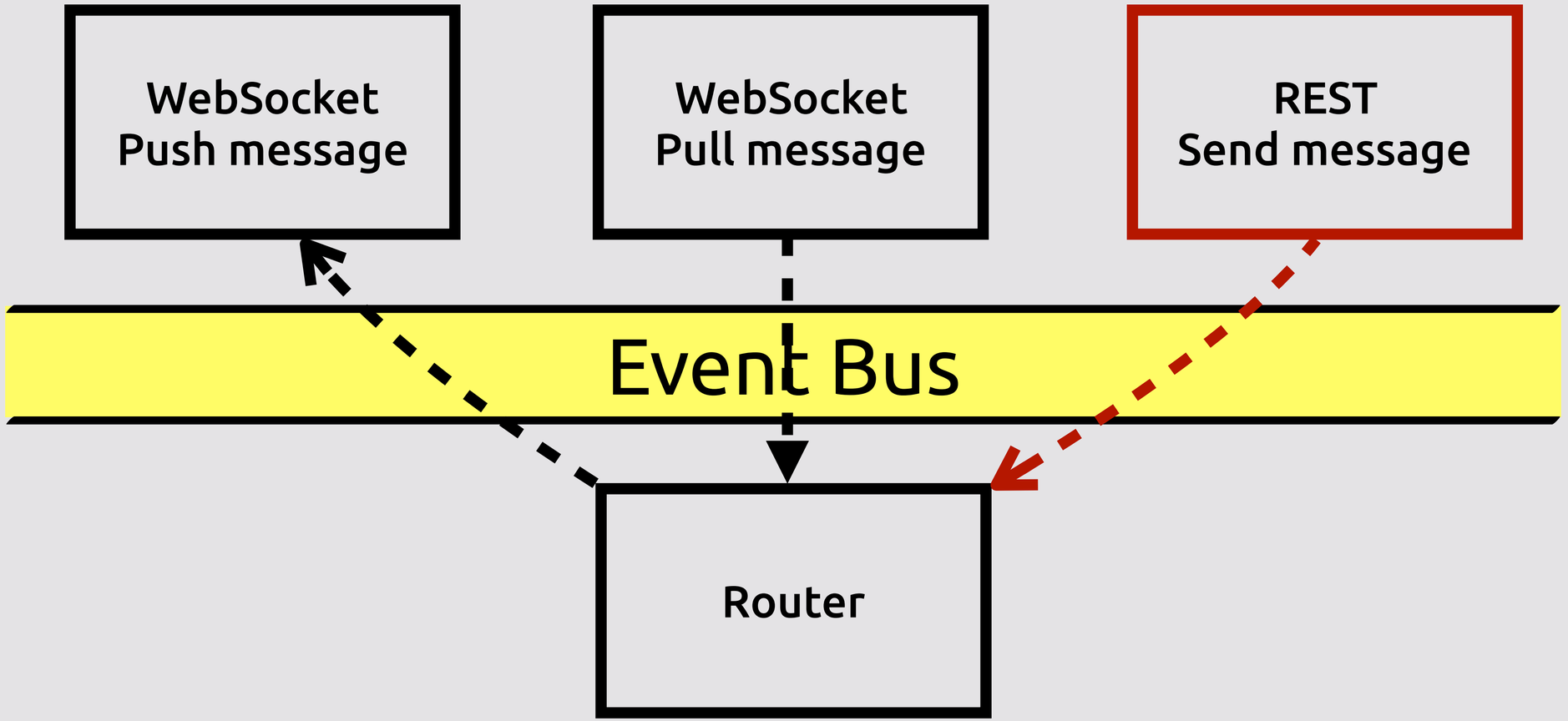
Научились

- Первое приложение
- WebSocket
- Кластеризация

Общая шина v2



Общая шина v2



Rest-клиент

```
curl \
-H "Content-Type: application/json" \
-X POST -d \
'{"text":"Message","address":"token"}' \
http://HOST/sendMessage
```

Отправка сообщений по REST


```
public class RestServerVerticle extends AbstractVerticle {
    @Override
    public void start() {
    }
}
```

Отправка сообщений по REST


```
public class RestServerVerticle extends AbstractVerticle {
    @Override
    public void start() {
    }
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
    .handler(request -> {
      vertx.eventBus().send("router", request.getBodyAsString());
      request.response().end("ok");
    });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

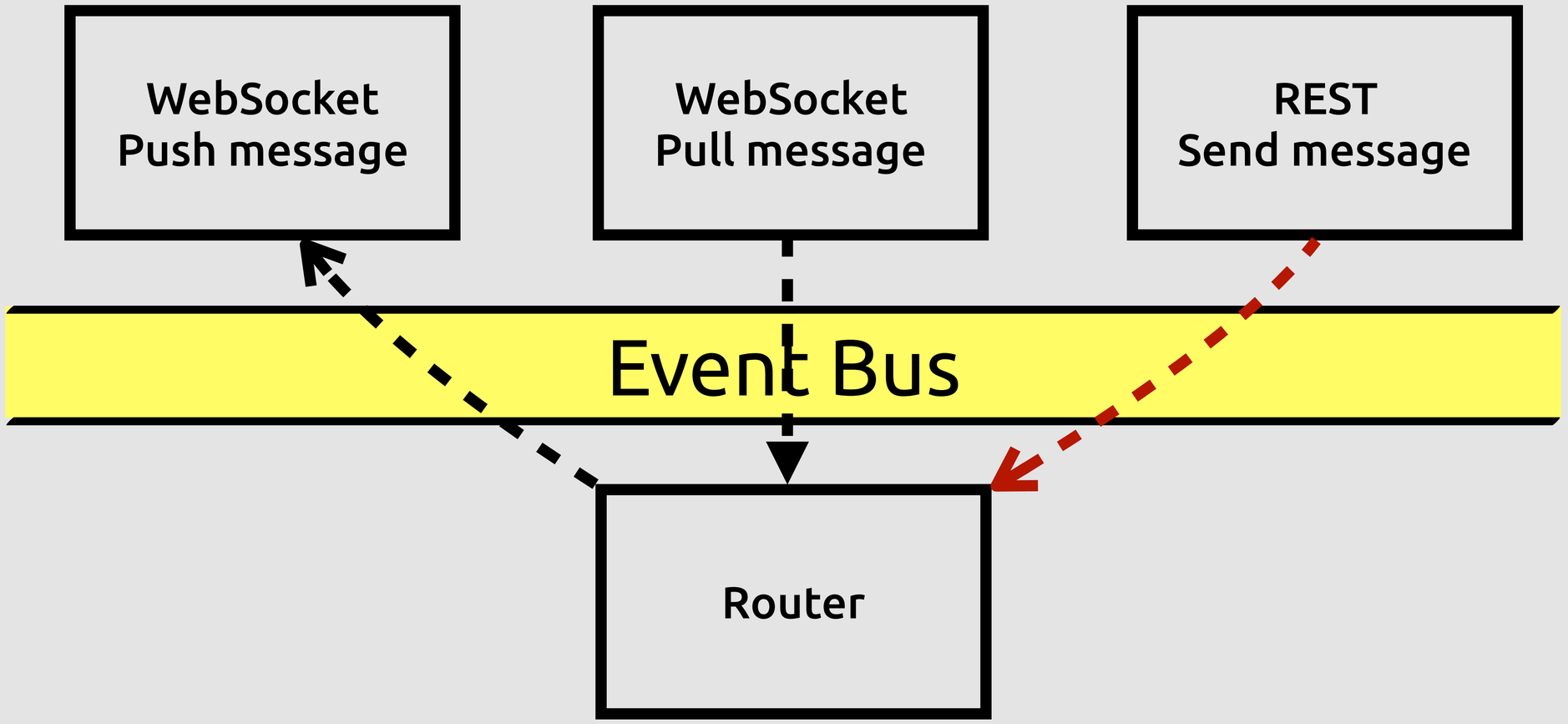
Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST



Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
@Override
public void start() {
  HttpServer httpServer = vertx.createHttpServer();
  Router httpRouter = Router.router(vertx);
  httpRouter.route().handler(BodyHandler.create());
  httpRouter.post("/sendMessage")
   .handler(request -> {
     vertx.eventBus().send("router", request.getBodyAsString());
     request.response().end("ok");
   });
  httpServer.requestHandler(httpRouter::accept);
  httpServer.listen(8081);
}
```

Отправка сообщений по REST


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chat-rest-server
  replicas: 1
# . . .
    spec:
      containers:
      - name: chat-rest-server
        image: dzx912/chat-rest-server:1
        ports:
        - containerPort: 8081
```

Отправка сообщений по REST


```
---
apiVersion: v1
kind: Service
metadata:
  name: chat-rest-server
  labels:
    run: chat-rest-server
spec:
  selector:
    run: chat-rest-server
  ports:
  - name: http
    port: 8081
    nodePort: 30081
  type: LoadBalancer
```

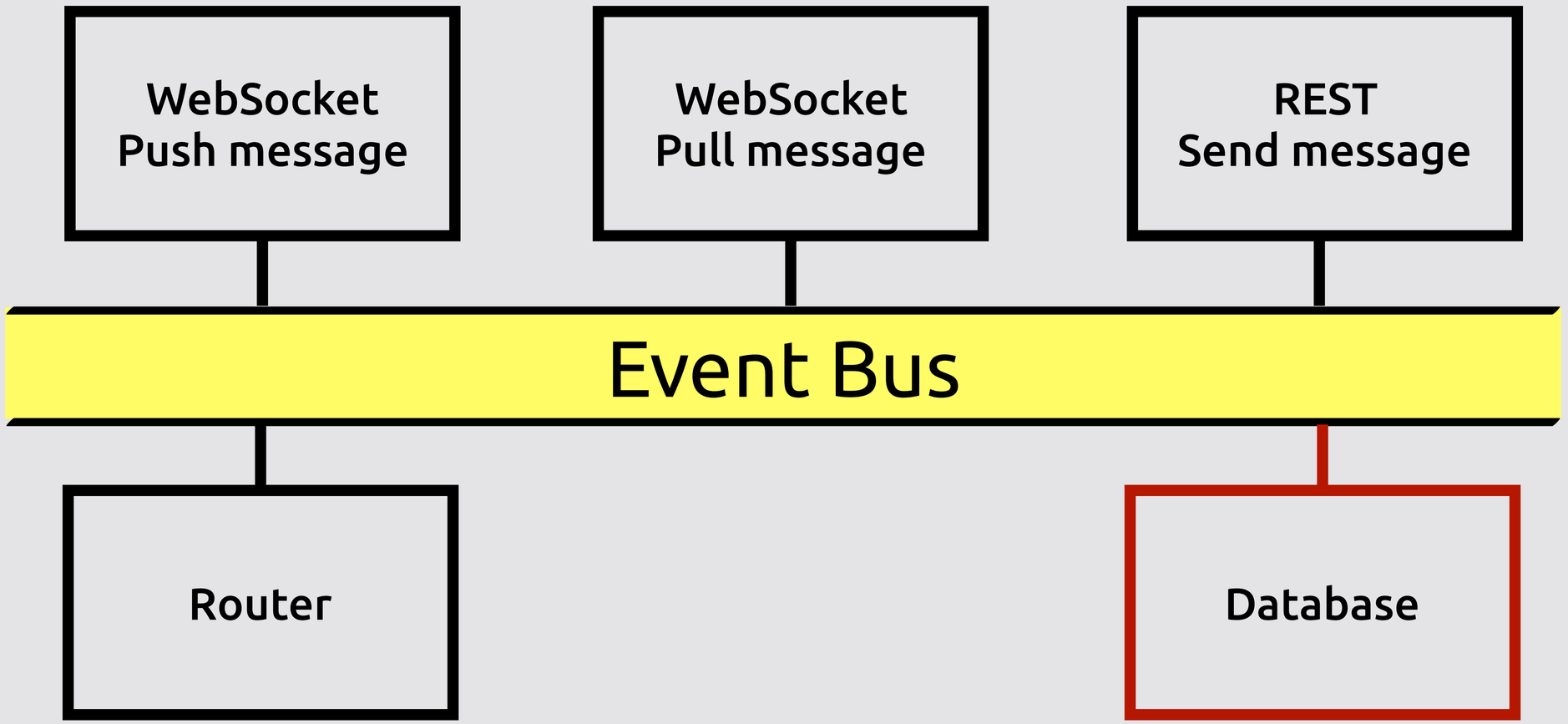
Отправка сообщений по REST

Demo

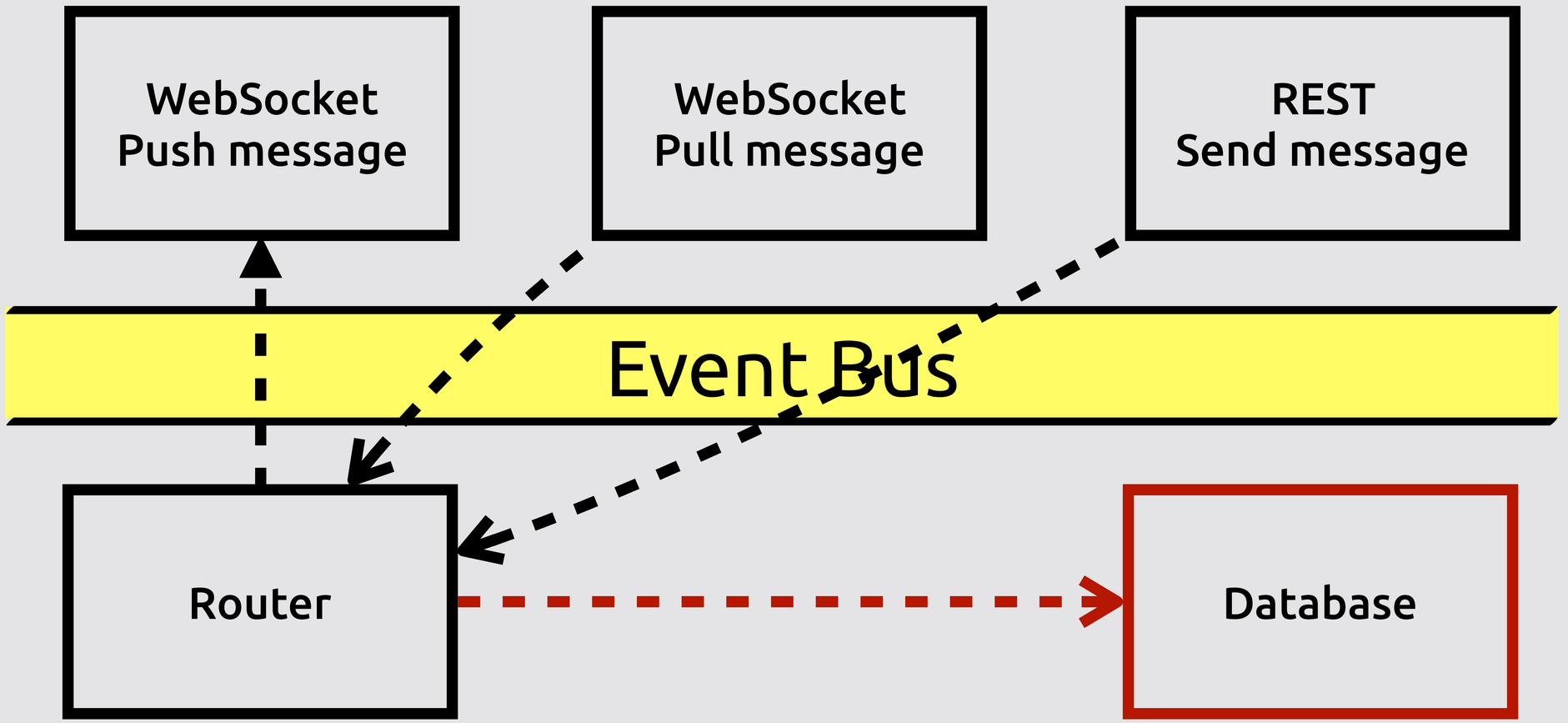
Научились

- Первое приложение
- WebSocket
- Кластеризация
- REST

Общая шина v3



Общая шина v3



MongoDB


```
public class RouterVerticle extends AbstractVerticle {
 void router(Message<String> message) {
  Data data = Json.decodeValue(message.body(), Data.class);
  vertx.eventBus().publish(
   "/token/" + data.getAddress(), message.body());

  vertx.eventBus().send("database.save", message.body());
 }
}
```

MongoDB


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo
# ...
  replicas: 1
    spec:
      containers:
      - name: mongo
        image: mongo
        ports:
        - containerPort: 27017
```

MongoDB


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo
# ...
  replicas: 1
    spec:
      containers:
      - name: mongo
        image: mongo
        ports:
        - containerPort: 27017
```

MongoDB


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo
# ...
  replicas: 1
    spec:
      containers:
      - name: mongo
        image: mongo
        ports:
        - containerPort: 27017
```

MongoDB


```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo
# ...
  replicas: 1
    spec:
      containers:
      - name: mongo
        image: mongo
        ports:
        - containerPort: 27017
```

MongoDB


```
---
apiVersion: v1
kind: Service
metadata:
  name: mongo
  labels:
    run: mongo
spec:
  selector:
    run: mongo
  ports:
  - port: 27017
    nodePort: 30002
  type: LoadBalancer
```

MongoDB


```
---
apiVersion: v1
kind: Service
metadata:
  name: mongo
  labels:
    run: mongo
spec:
  selector:
    run: mongo
  ports:
  - port: 27017
    nodePort: 30002
  type: LoadBalancer
```

MongoDB


```
---
apiVersion: v1
kind: Service
metadata:
  name: mongo
  labels:
    run: mongo
spec:
  selector:
    run: mongo
  ports:
  - port: 27017
    nodePort: 30002
  type: LoadBalancer
```

MongoDB


```
---
apiVersion: v1
kind: Service
metadata:
  name: mongo
  labels:
    run: mongo
spec:
  selector:
    run: mongo
  ports:
  - port: 27017
    nodePort: 30002
  type: LoadBalancer
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

MongoDB


```
---
apiVersion: v1
kind: Service
metadata:
  name: mongo
  labels:
    run: mongo
spec:
  selector:
    run: mongo
  ports:
  - port: 27017
    nodePort: 30002
  type: LoadBalancer
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

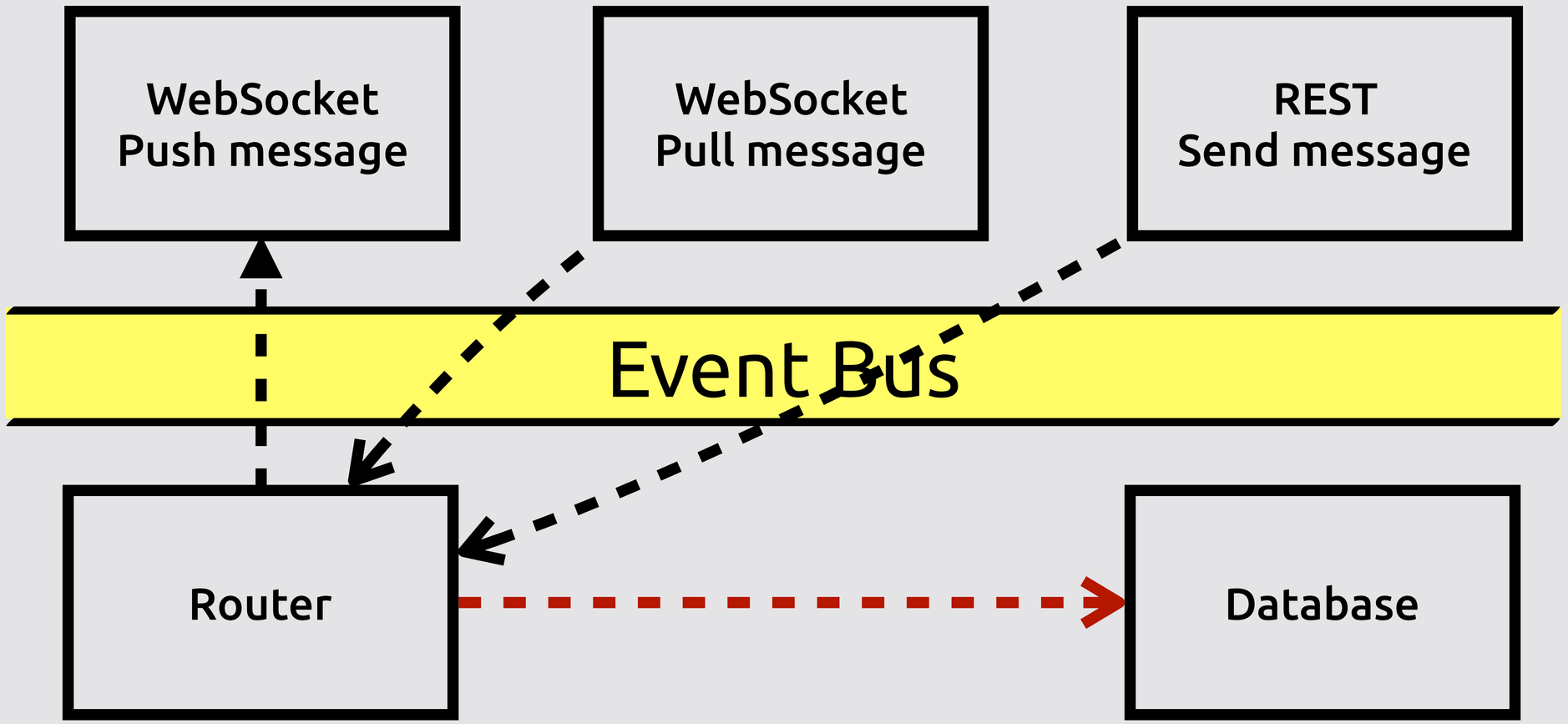
MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
```

MongoDB


WebSocket
Push message
WebSocket
Pull message
REST
Send message
Event Bus
Router
Database

MongoDB


```
public class MongoDbVerticle extends AbstractVerticle {
  private MongoClient client;
  @Override
  public void start() {
    String uri = "mongodb://mongo:27017";
    client = MongoClient.createShared(vertx, new JsonObject()
        .put("connection_string", uri)
        .put("db_name", "my_DB"));
    vertx.eventBus().consumer("database.save", this::saveDb);
  }
}
void saveDb(Message<String> message) {
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

MongoDB


```
void createWebSocketServer(ServerWebSocket wsServer) {
 wsServer.frameHandler(wsFrame -> {
  vertx.eventBus().send(
  "router", wsFrame.textData());
 });
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

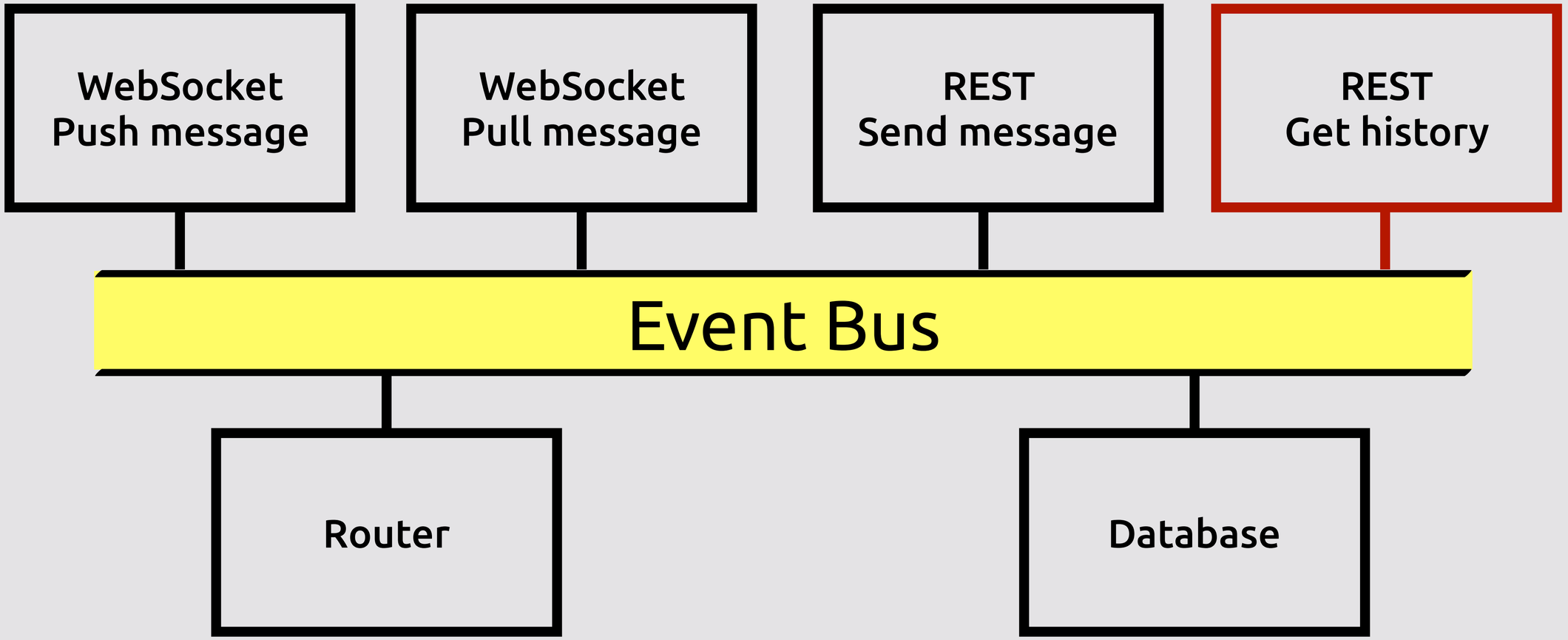
MongoDB


```
void saveDb(Message<String> message) {
  client.insert("message",
    new JsonObject(message.body()), this::handlerSaved);
}

void handlerSaved(AsyncResult<String> stringAsyncResult) {
  if (stringAsyncResult.succeeded()) {
    System.out.println("MongoDB save: "
      + stringAsyncResult.result());
  } else {
    System.out.println("ERROR MongoDB: "
      + stringAsyncResult.cause());
  }
}
```

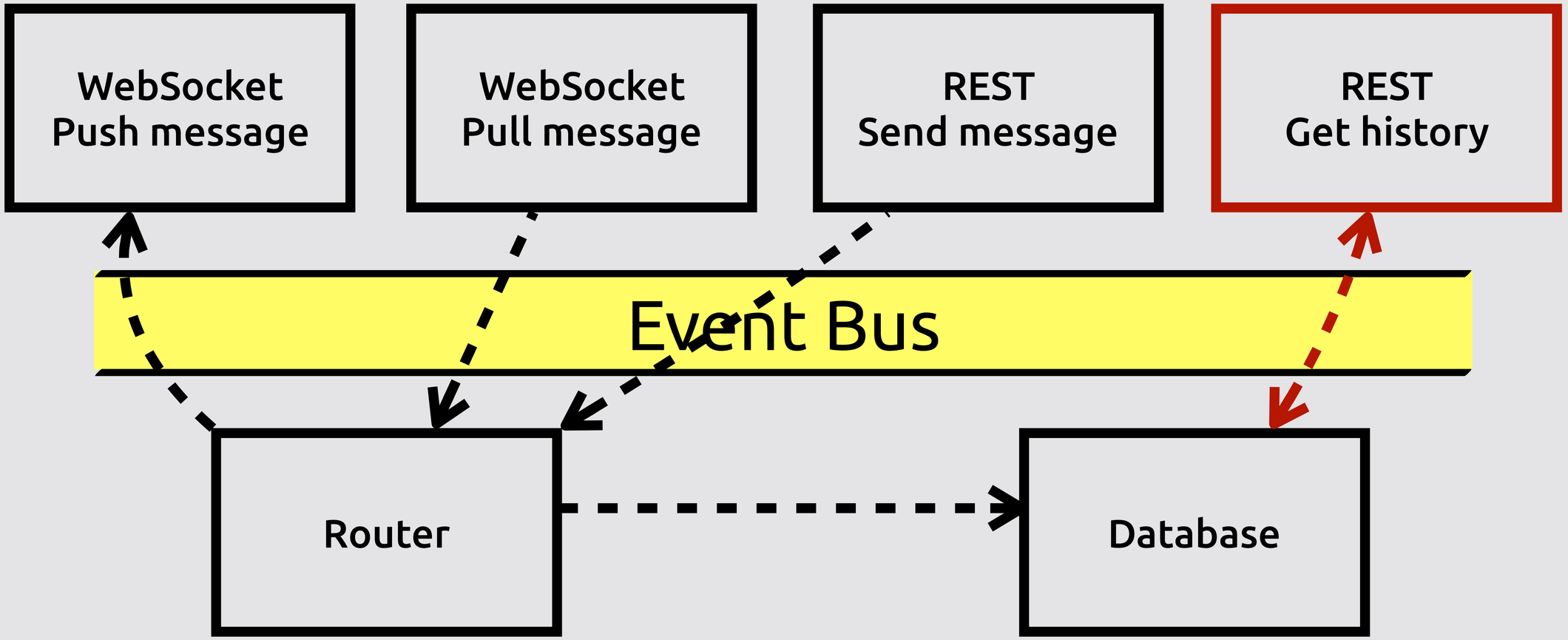
Научились

- Первое приложение
- WebSocket
- Кластеризация
- REST
- База данных

Общая шина v4



Общая шина v4



Rest-клиент

```
curl http://HOST/message/ADDRESS
```

Прием сообщений по REST


```
public class RestServerVerticle extends AbstractVerticle {
    @Override
    public void start() {
    }
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

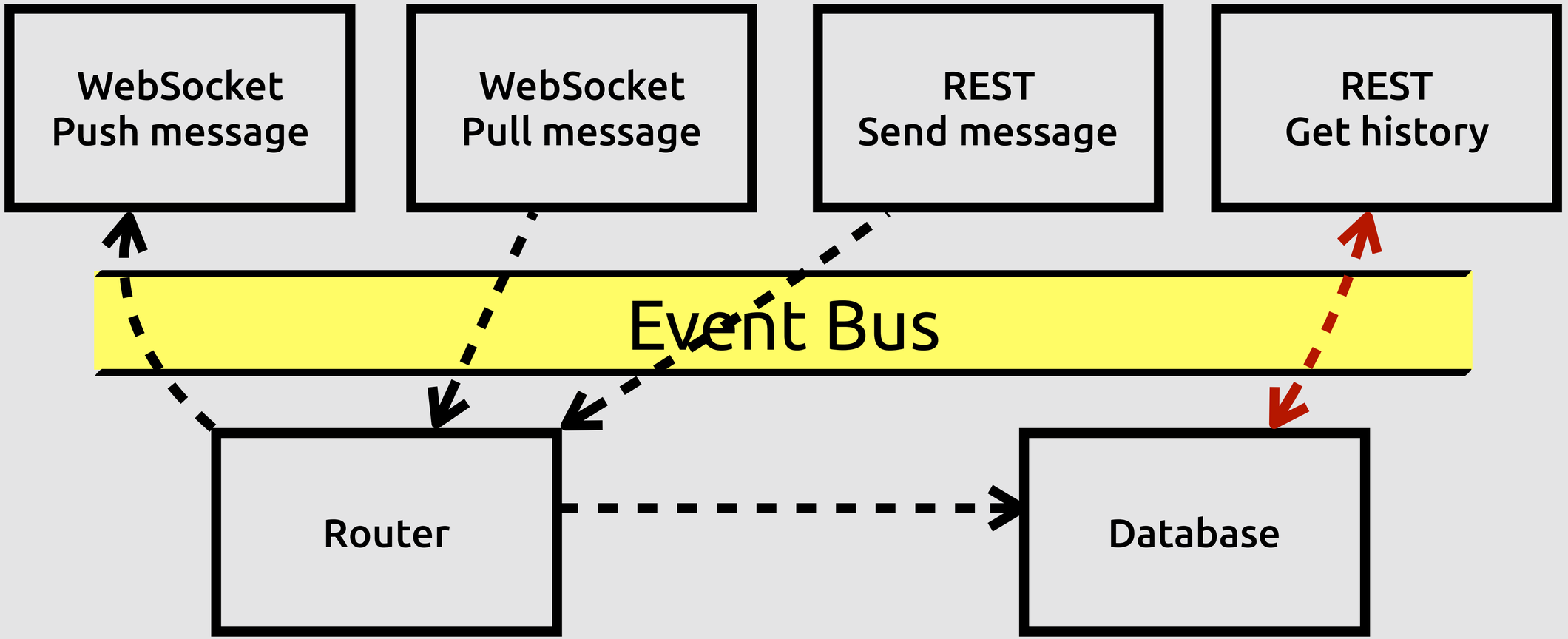
Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST



Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST


```
public class MongoDbVerticle extends AbstractVerticle {
 @Override
  public void start() {
  // . . .
  vertx.eventBus().consumer("getHistory", this::getHistory);
  }
}
void getHistory(Message<String> message) {
    client.find("message",
            new JsonObject().put("address", message.body()),
            result -> {
                String history = Json.encode(result.result());
                message.reply(history);
  });
}
```

Прием сообщений по REST


```
public class MongoDbVerticle extends AbstractVerticle {
 @Override
  public void start() {
  // . . .
  vertx.eventBus().consumer("getHistory", this::getHistory);
  }
}
void getHistory(Message<String> message) {
    client.find("message",
            new JsonObject().put("address", message.body()),
            result -> {
                String history = Json.encode(result.result());
                message.reply(history);
  });
}
```

Прием сообщений по REST


```
public class MongoDbVerticle extends AbstractVerticle {
 @Override
  public void start() {
  // . . .
  vertx.eventBus().consumer("getHistory", this::getHistory);
  }
}
void getHistory(Message<String> message) {
    client.find("message",
            new JsonObject().put("address", message.body()),
            result -> {
                String history = Json.encode(result.result());
                message.reply(history);
  });
}
```

Прием сообщений по REST


```
public class MongoDbVerticle extends AbstractVerticle {
 @Override
  public void start() {
  // . . .
  vertx.eventBus().consumer("getHistory", this::getHistory);
  }
}
void getHistory(Message<String> message) {
    client.find("message",
            new JsonObject().put("address", message.body()),
            result -> {
                String history = Json.encode(result.result());
                message.reply(history);
  });
}
```

Прием сообщений по REST


```
public class MongoDbVerticle extends AbstractVerticle {
 @Override
  public void start() {
  // . . .
  vertx.eventBus().consumer("getHistory", this::getHistory);
  }
}
void getHistory(Message<String> message) {
    client.find("message",
            new JsonObject().put("address", message.body()),
            result -> {
                String history = Json.encode(result.result());
                message.reply(history);
  });
}
```

Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

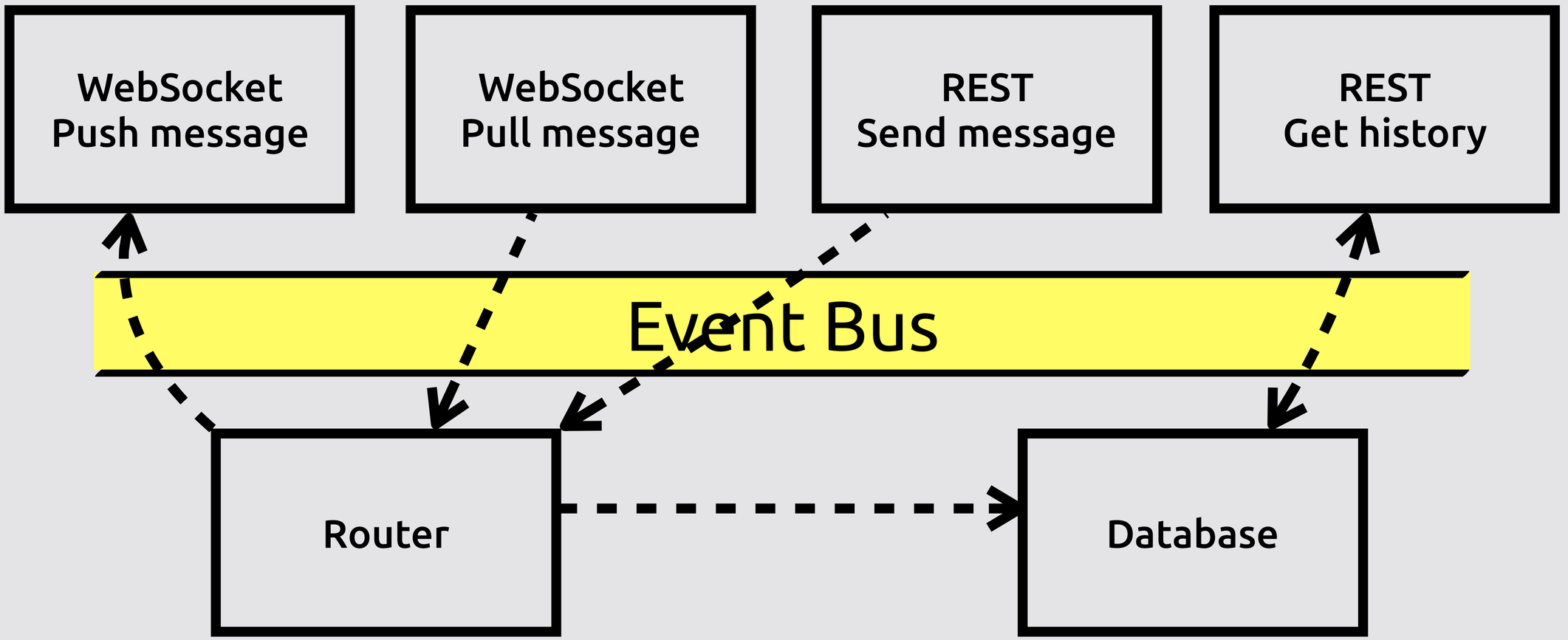
Прием сообщений по REST


```
@Override
public void start() {
// . . .
httpRouter.get("/message/:address")
  .handler(request -> {
    String address = request.request().getParam("address");
    vertx.eventBus().send("getHistory", address, result -> {
      String history = result.result().body().toString();
      request.response().end(history);
});});});
// . . .
}
```

Прием сообщений по REST

Demo

Общая шина


WebSocket
Push message
WebSocket
Pull message
REST
Send message
REST
Get history
Event Bus
Router
Database

Продумывайте маршруты

Чистим за собой

```
kubectl delete -f chat.yaml
```

Какие «бонусы» дает Vert.x

- Быстрый отклик
- Простой параллелизм
- Легкое масштабирование

Когда не стоит использовать Vert.x

- Выполнить все операции или ни одной
- Нужен четкий порядок действий
- Не готовы изучить новую технологию

Ленок Антон

Серверная разработка

Проект: Диалоги
в Сбербанк Онлайн

AILenok.SBT@sberbank.ru

URL: https://github.com/dzx912/jpoint-2018-vertx