GridGain

Microservices Architecture on top of Apache Ignite

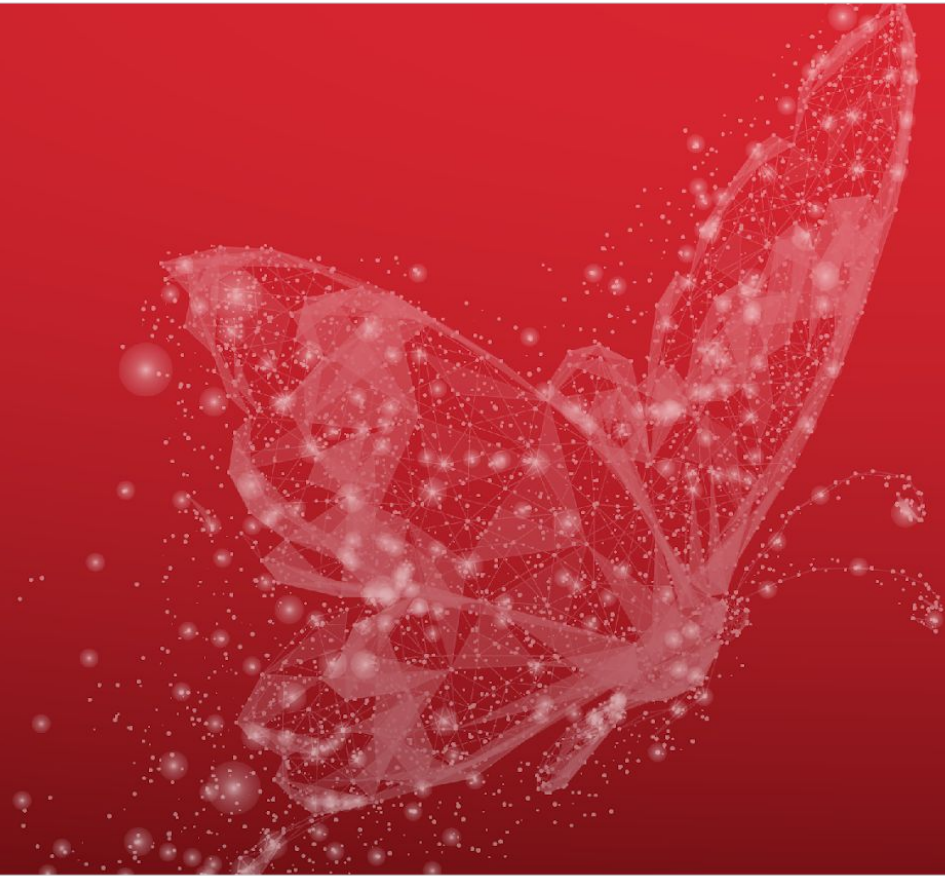
Denis Mekhanikov

November 6, 2019

Plan

- What services are?
- How to use them in your applications?
- How do they work internally?

Services: Overview

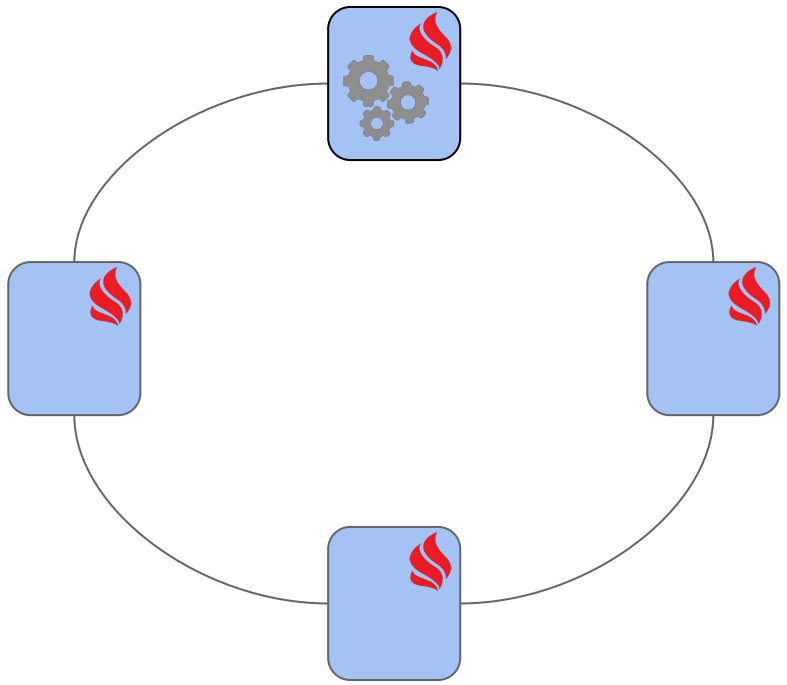


Interface

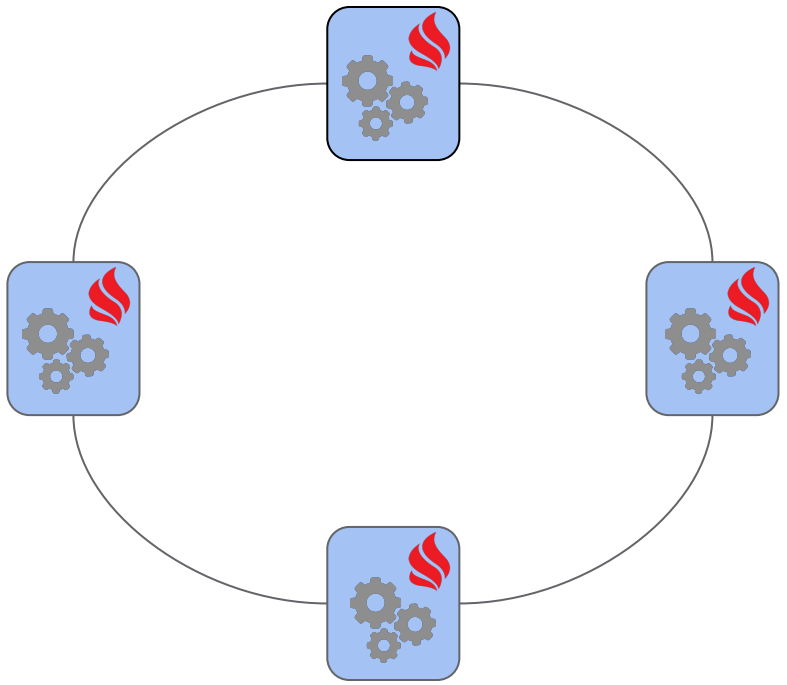
```
public interface Service extends Serializable {
    void init(ServiceContext ctx) throws Exception;

    void execute(ServiceContext ctx) throws Exception;

    void cancel(ServiceContext ctx);
}
```

Deployment modes

Cluster singleton

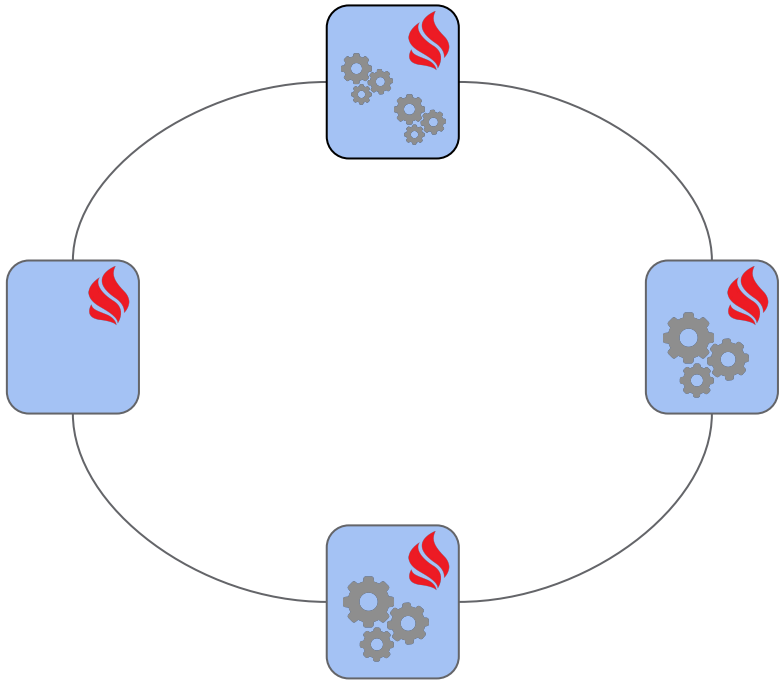


Node singleton



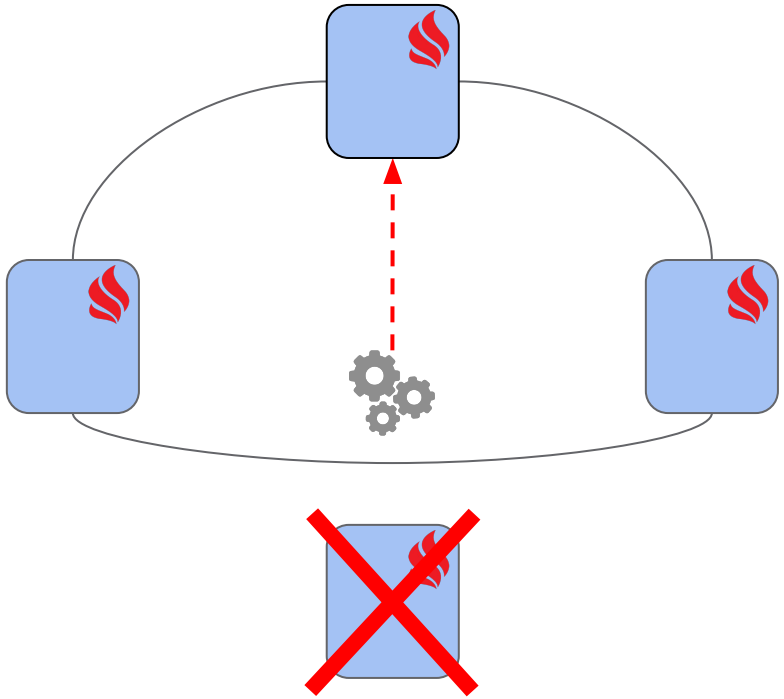
Flexible deployment mode tuning

- **TotalCount** – required number of instances in the whole cluster
- **MaxPerNodeCount** – maximum number of instances per node



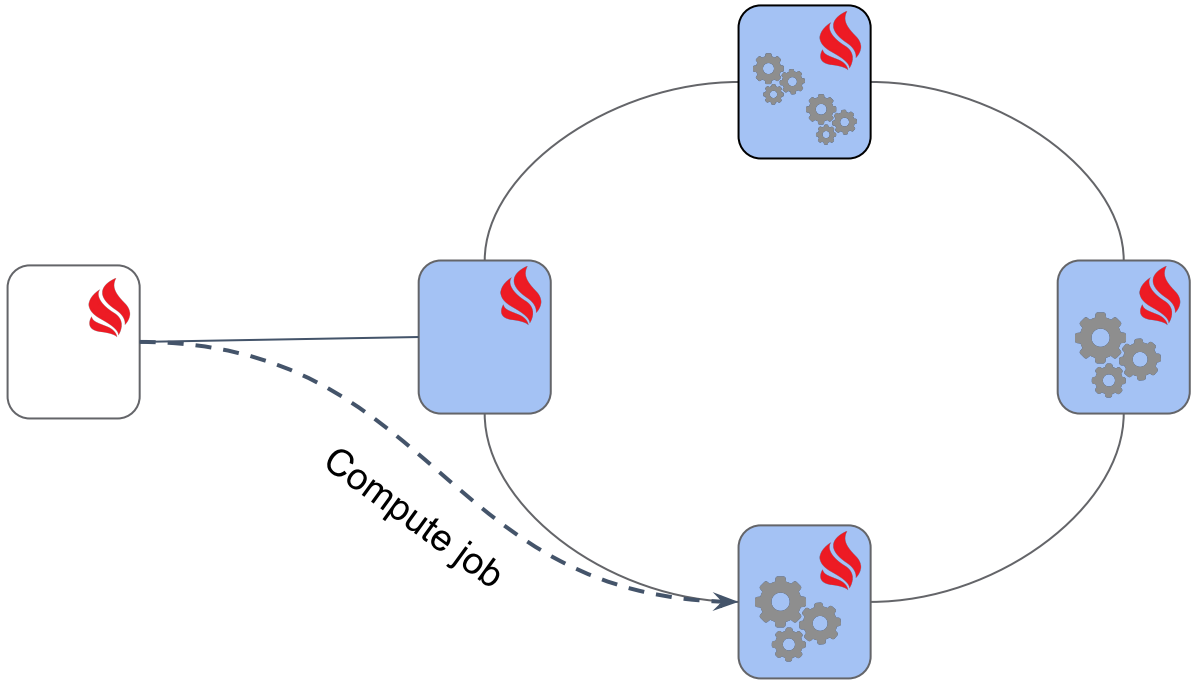
Failover

When the cluster topology changes,
missing services are deployed
to match deployment mode
requirements



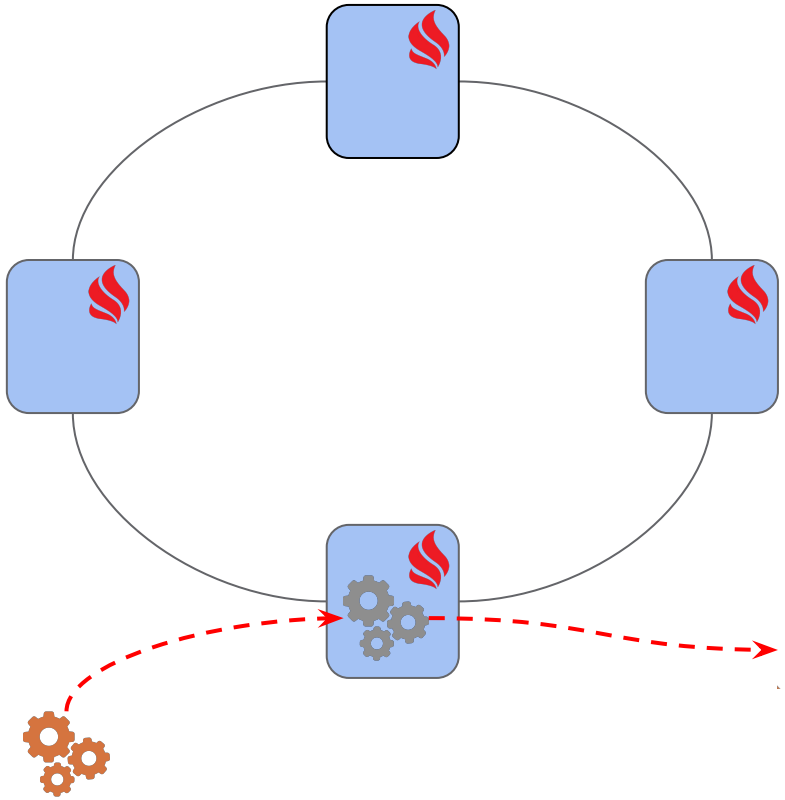
Remote procedure call

Call methods of services that are deployed remotely



Hot redeployment

Update implementation of deployed services without restarting the grid



Example: SimpleMapService

https://github.com/apache/ignite/tree/master/examples

Example: SimpleMapService (1/4)

```
public interface SimpleMapService<K, V> {
    void put(K key, V val);

    V get(K key);

    void clear();

    int size();
}
```

Example: SimpleMapService (2/4)

```
public class SimpleMapServiceImpl<K, V> implements Service, SimpleMapService<K, V> {
    @IgniteInstanceResource
    private Ignite ignite;

    private IgniteCache<K, V> cache;

    @Override public void init(ServiceContext ctx) throws Exception {
        cache = ignite.getOrCreateCache(
                    new CacheConfiguration<>(ctx.name()));
    }

    @Override public void cancel(ServiceContext ctx) {
        ignite.destroyCache(ctx.name());
    }
    // ...
}
```

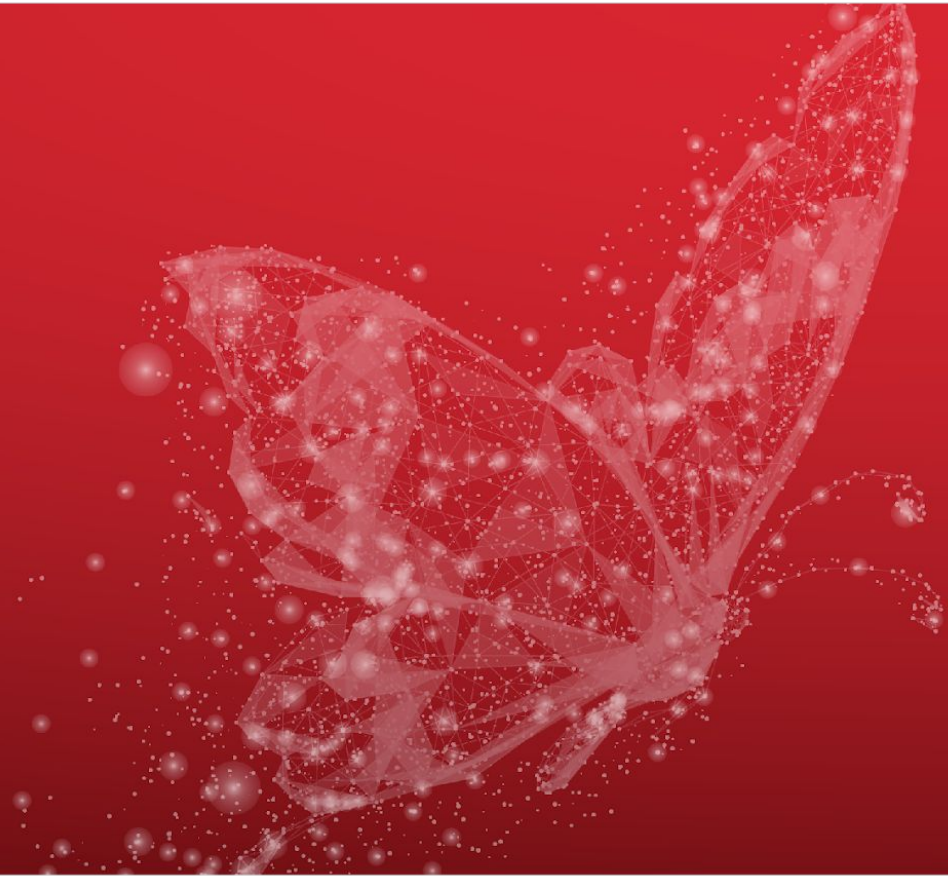
Example: SimpleMapService (3/4)

```
@Override public void put(K key, V val) {
    cache.put(key, val);
}

@Override public V get(K key) {
    return cache.get(key);
}

@Override public void clear() {
    cache.clear();
}

@Override public int size() {
    return cache.size();
}
```

Example: SimpleMapService (4/4)

```
SimpleMapService<Integer, String> mapSvc =
    ignite.services().serviceProxy(
        "map-service",
        SimpleMapService.class,
        true);

mapSvc.put(100, "100");
int size = mapSvc.size();
```

Example: Search Engine



https://github.com/dmekhanikov/ignite-text-search

Entities

```
public class Document {
    UUID id;
    String content;
    long timeAdded;
}
```

```
public class Match implements Comparable<Match> {
    Document doc;
    int num;
}
```

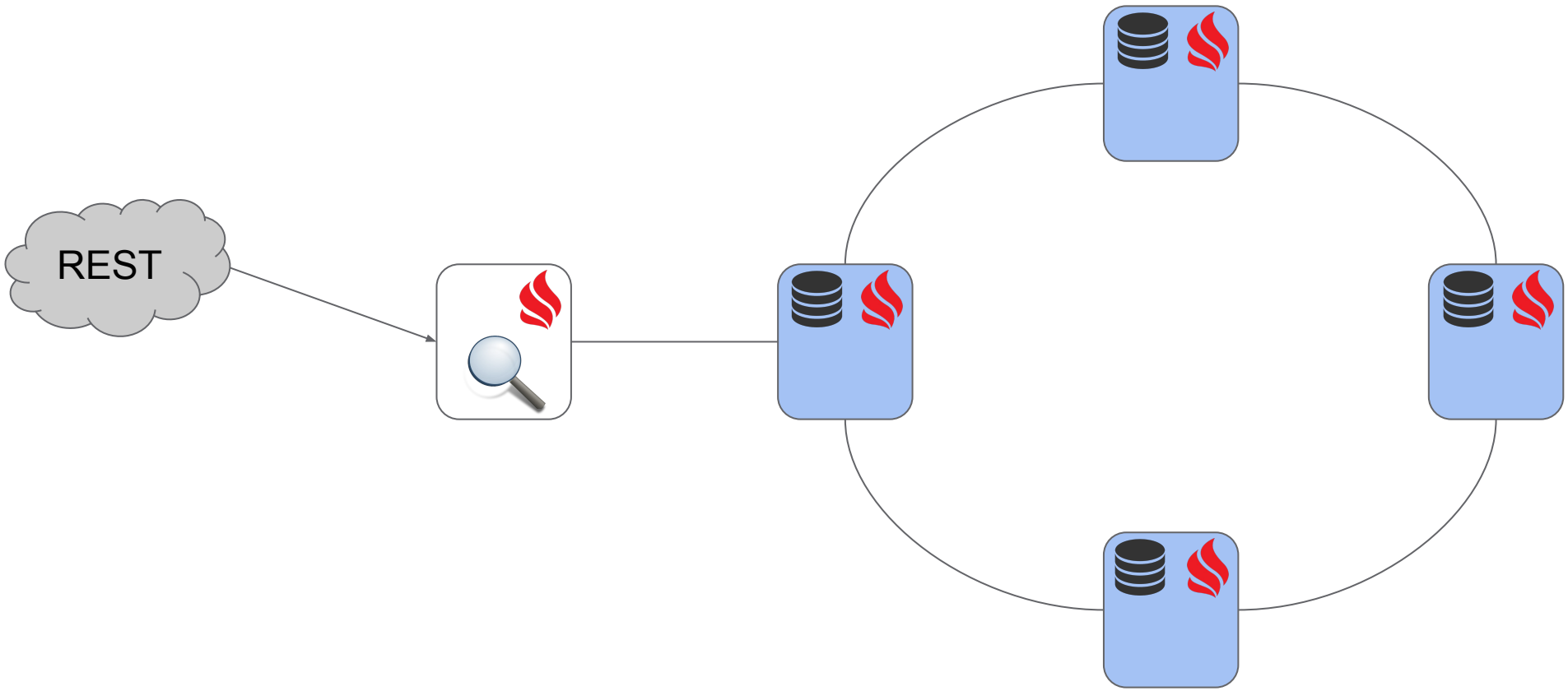
Adding documents

```
private IgniteCache<UUID, Document> documentsCache;

public Document add(String text) {
    UUID id = UUID.randomUUID();
    Document doc = new Document(id, text,
                                System.currentTimeMillis());
    documentsCache.put(id, doc);
    return doc;
}
```

Search

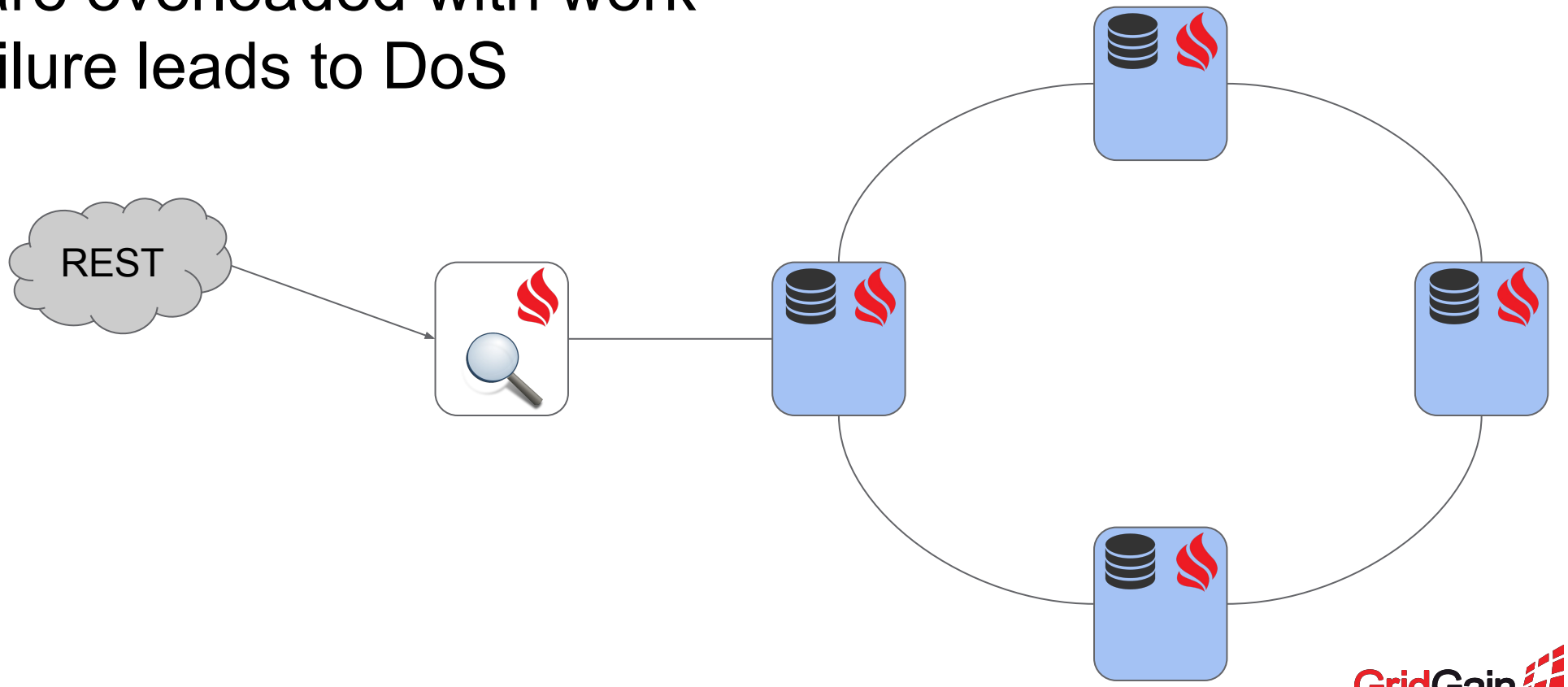
```
public List<Match> search(String qryText, int limit) {
    LinkedList<Match> result = new LinkedList<>();

    Query<Cache.Entry<UUID, Document>> qry = new ScanQuery<>(
                                                (k, v) -> v.getContent().contains(qryText));

    try (QueryCursor<Cache.Entry<UUID, Document>> cursor = documentsCache.query(qry)) {
        for (Cache.Entry<UUID, Document> e : cursor) {
            Document doc = e.getValue();
            int num = countOccurrencesOf(doc.getContent(), qryText);
            Match match = new Match(doc, num);
            insert(result, match, limit);
        }
    }

    return result;
}
```

Approach #1: client does everything



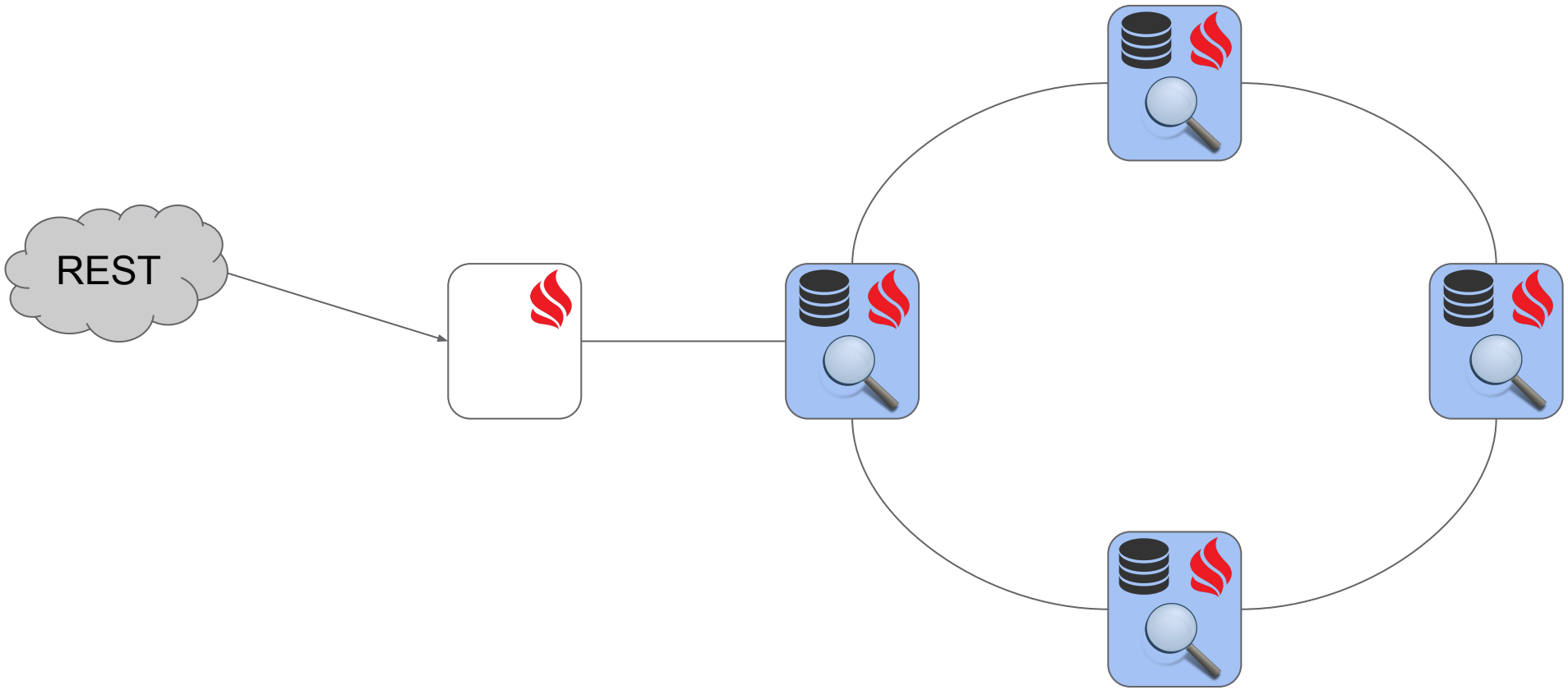
Issues of the approach #1

- Clients are too clever
- Clients are overloaded with work
- Client failure leads to DoS



Approach #2

- Load balancing



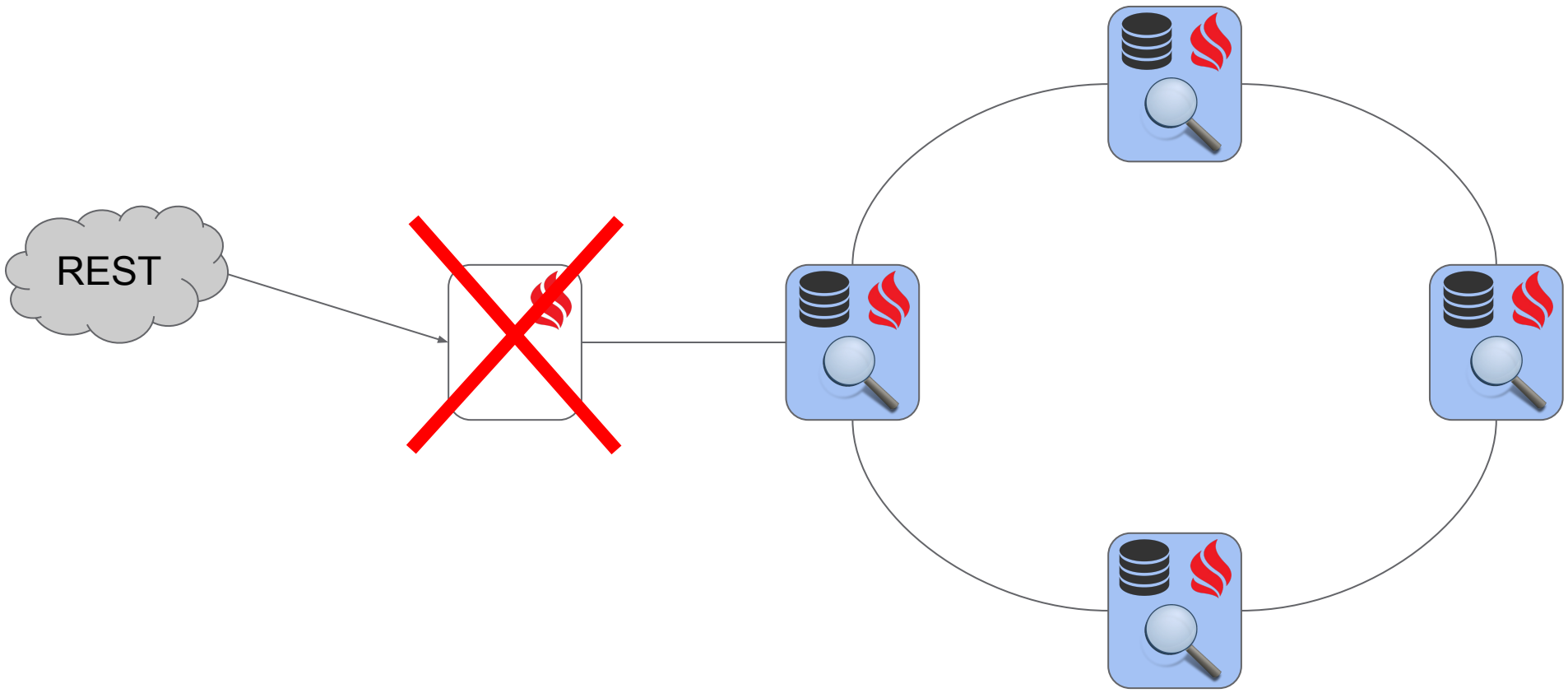
Approach #2: TextSearchService

```
public interface TextSearchService {
    Document add(String text);
    List<Document> addAll(List<String> lines);
    List<Match> search(String query, int limit);
}

public class TextSearchServiceImpl
            implements TextSearchService, Service {
    // Implementation.
}
```

Approach #2: deploying a service

```
ignite.services().deployNodeSingleton(
        "search-service",
        new TextSearchServiceImpl());
```

Approach #3: get rid of the client

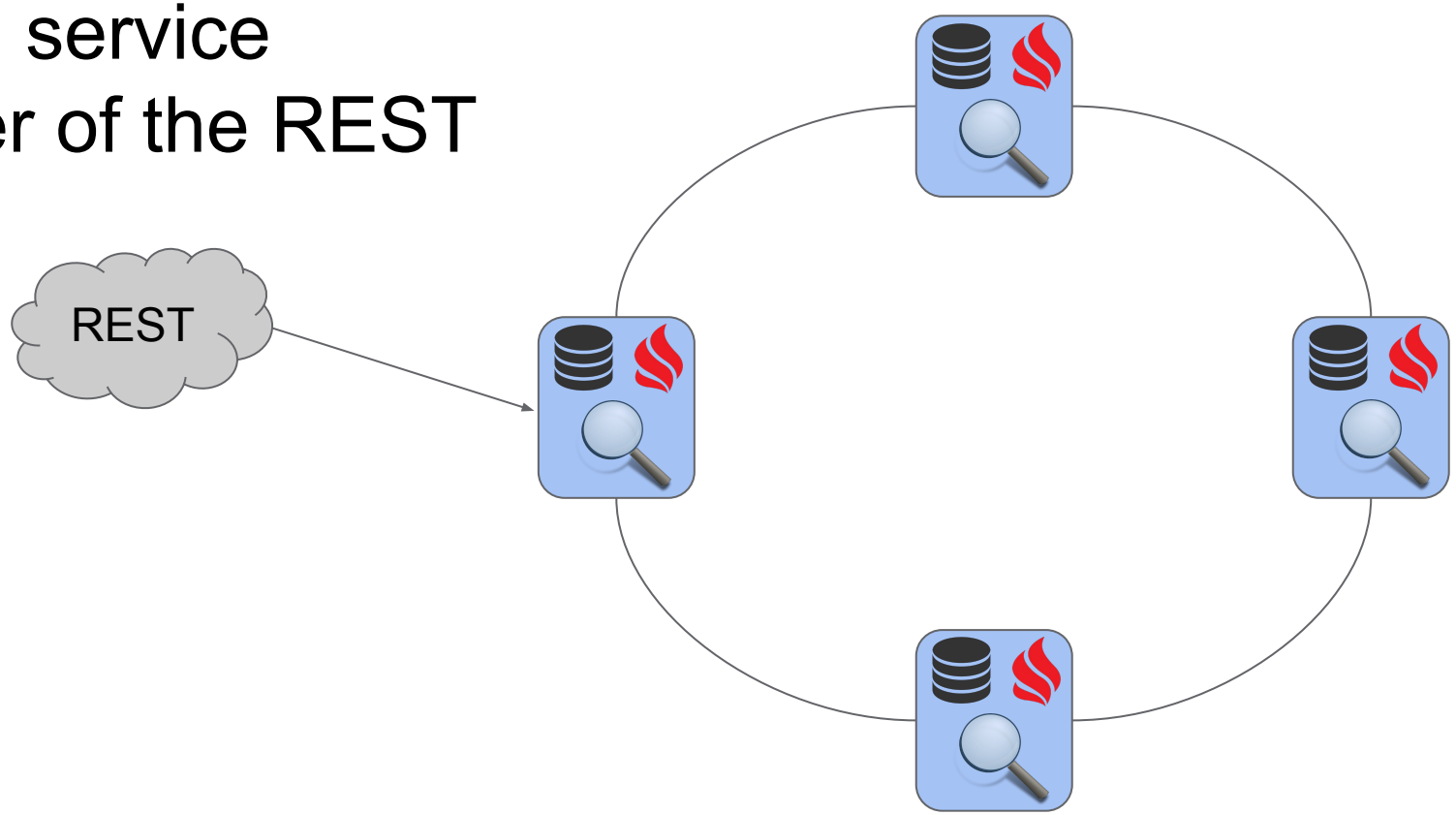


Approach #3: REST endpoint as a service

```
ignite.services().deployClusterSingleton(
        "rest-service",
        new RestEndpointService());
```

Approach #3

- A client node is not needed
- A cluster singleton service guarantees failover of the REST endpoint



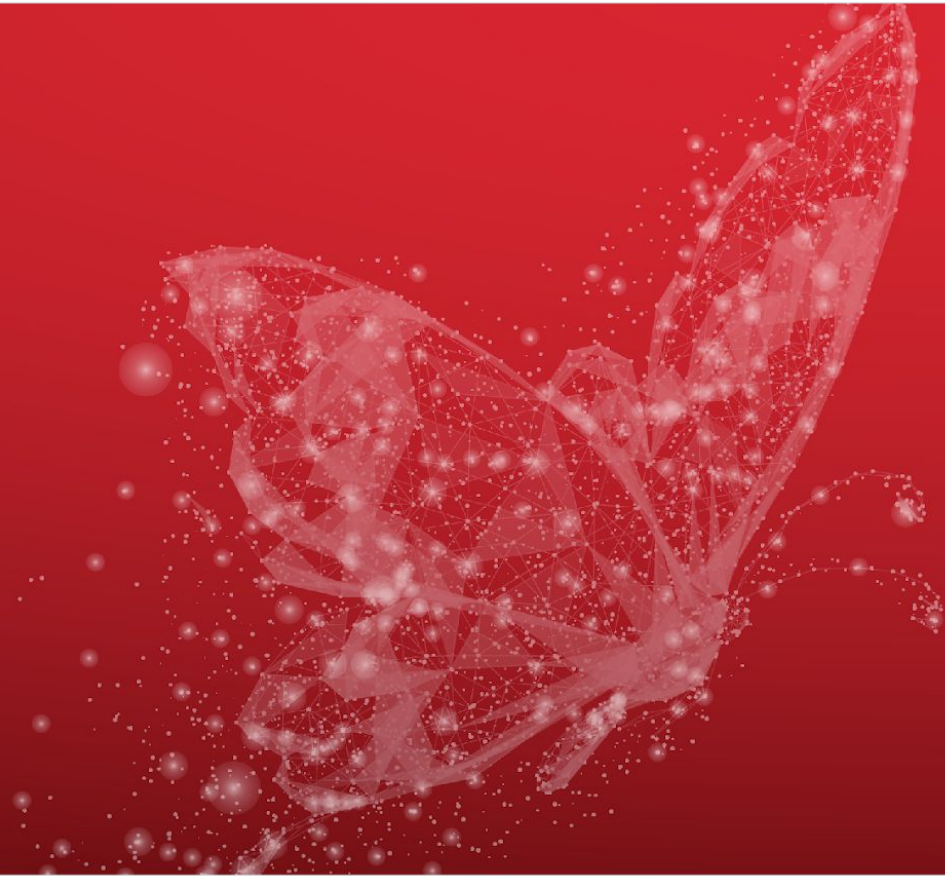
Approach #4: caching of results

```
private final Map<Request, LinkedList<Match>> localRequestCache =
        Collections.synchronizedMap(new LinkedHashMap<>());

public List<Match> search(String qryText, int limit) {
    Request req = new Request(qryText, limit, System.currentTimeMillis());
    LinkedList<Match> result = localRequestCache.get(req);
    if (result != null)
        return result;

    // ...

    if (localRequestCache.size() < CACHE_MAX_SIZE)
        localRequestCache.put(req, result);
    return result;
}
```

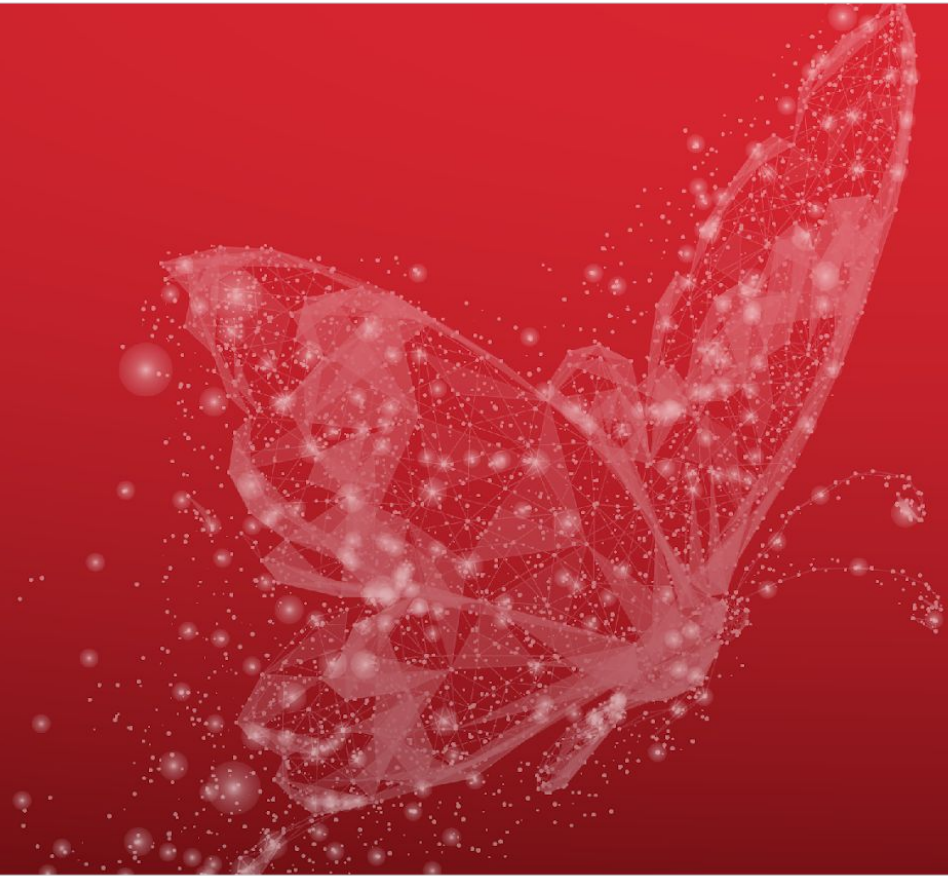
Approach #4: cleaning the cache

```
public void execute(ServiceContext serviceContext) throws InterruptedException {
    while (!serviceContext.isCancelled()) {
        synchronized (localRequestCache) {
            Iterator<Map.Entry<Request, LinkedList<Match>>> it =
                                                localRequestCache.entrySet().iterator();

            while (it.hasNext()) {
                Request req = it.next().getKey();
                if (req.getTimestamp() + CACHE_LIFE_TIME >= System.currentTimeMillis())
                    it.remove();
                else
                    break;
            }
        }

        Thread.sleep(CACHE_LIFE_TIME);
    }
}
```

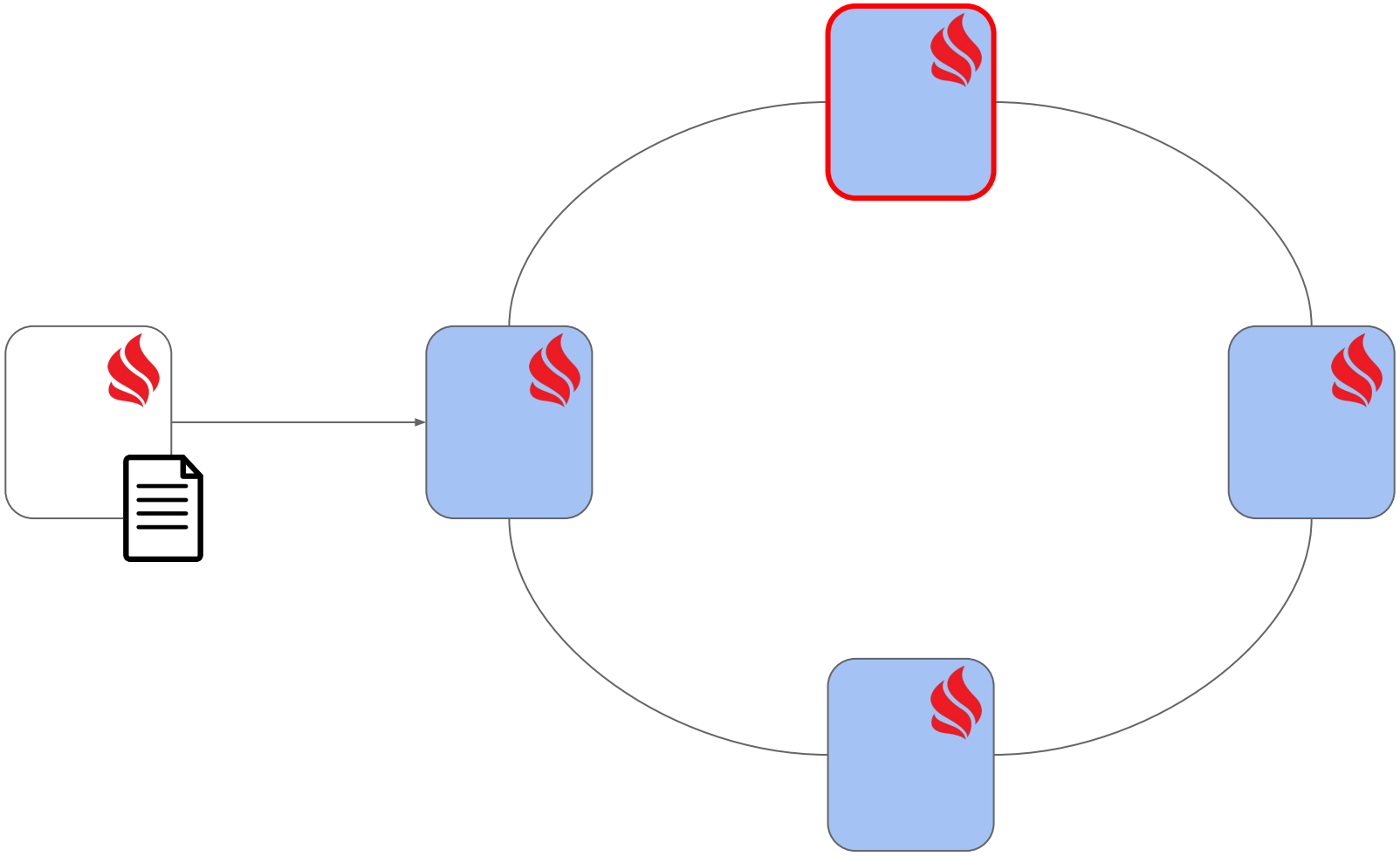
Demo



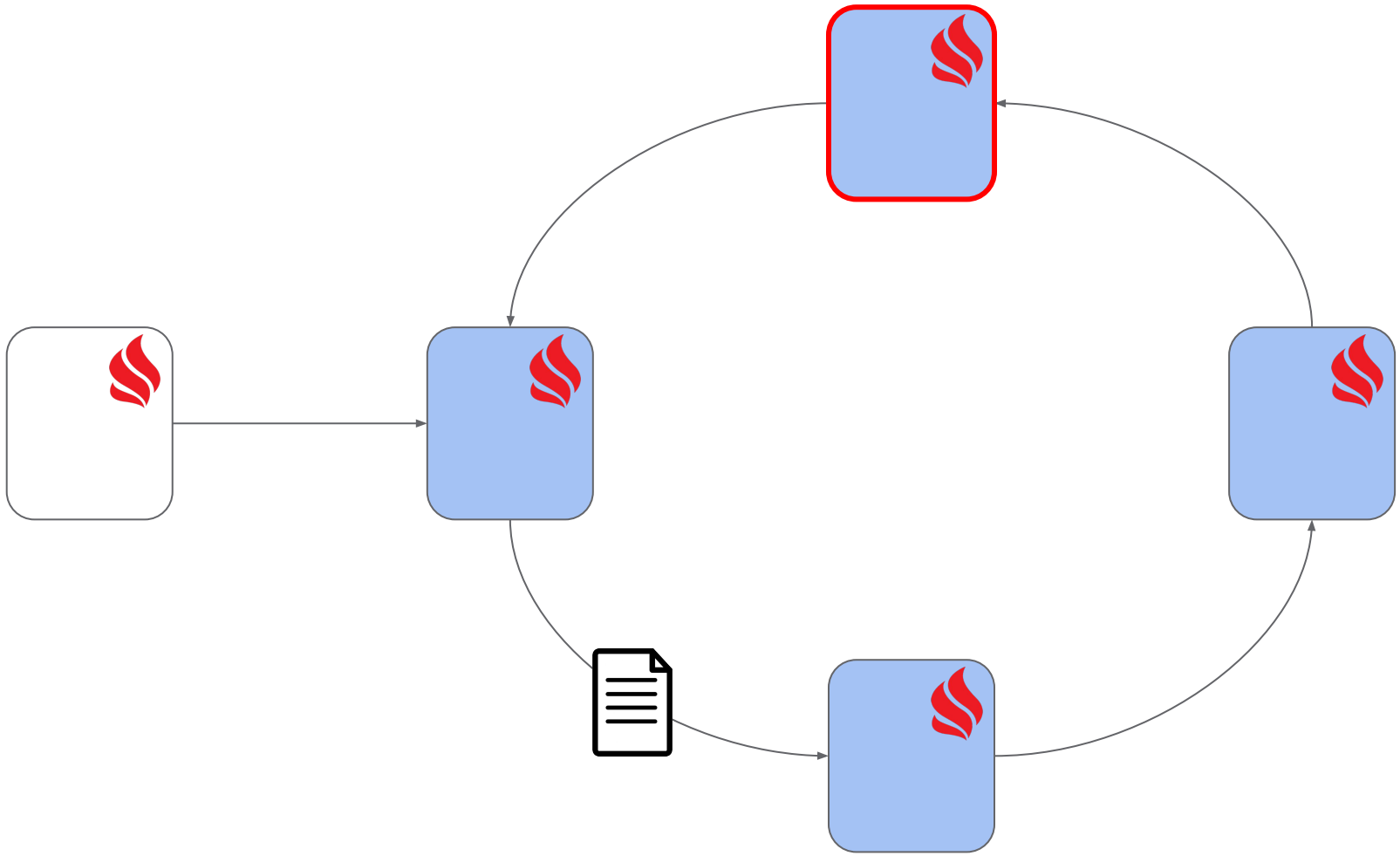
Services Under the Hood



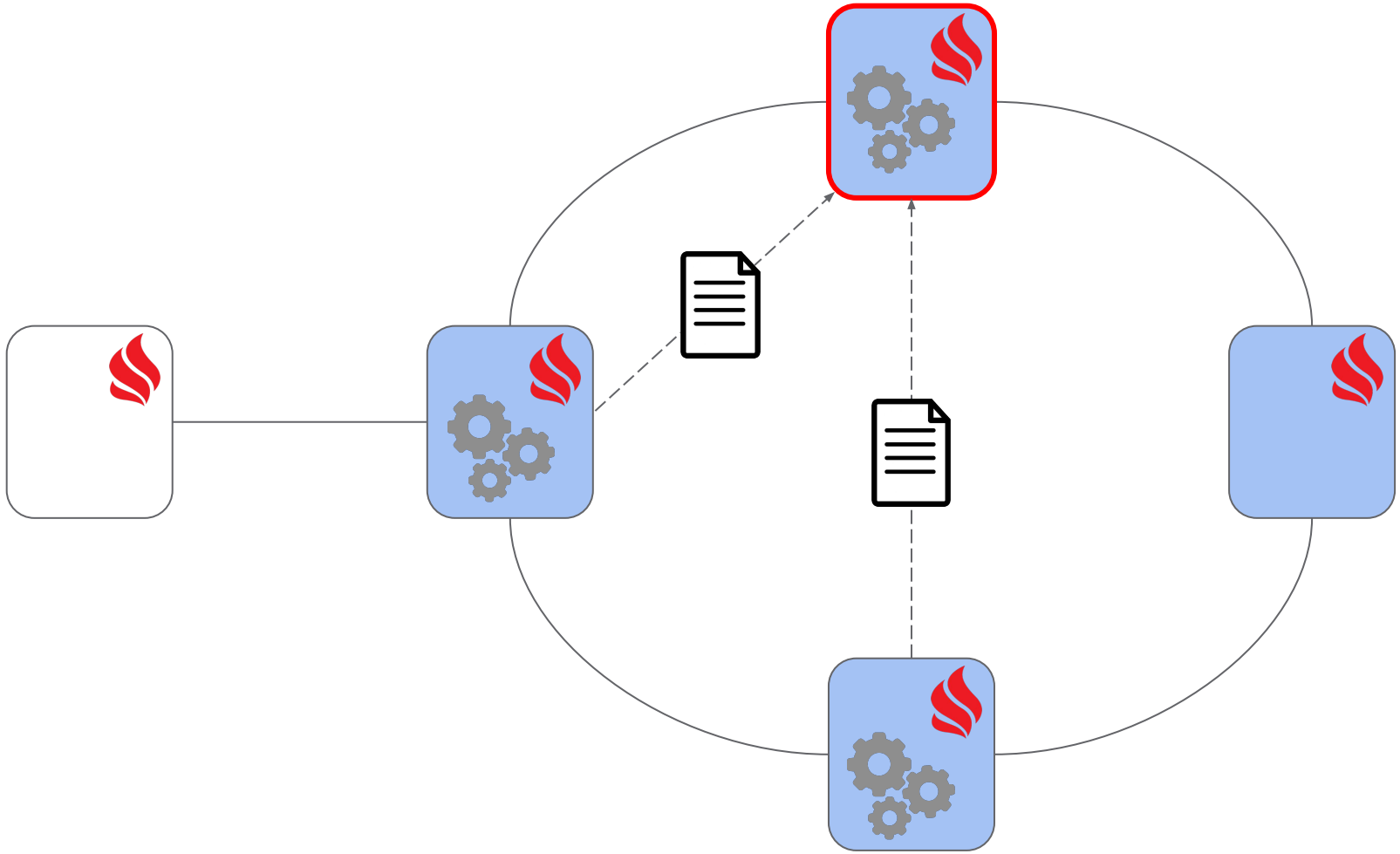
Deployment procedure (1/4)



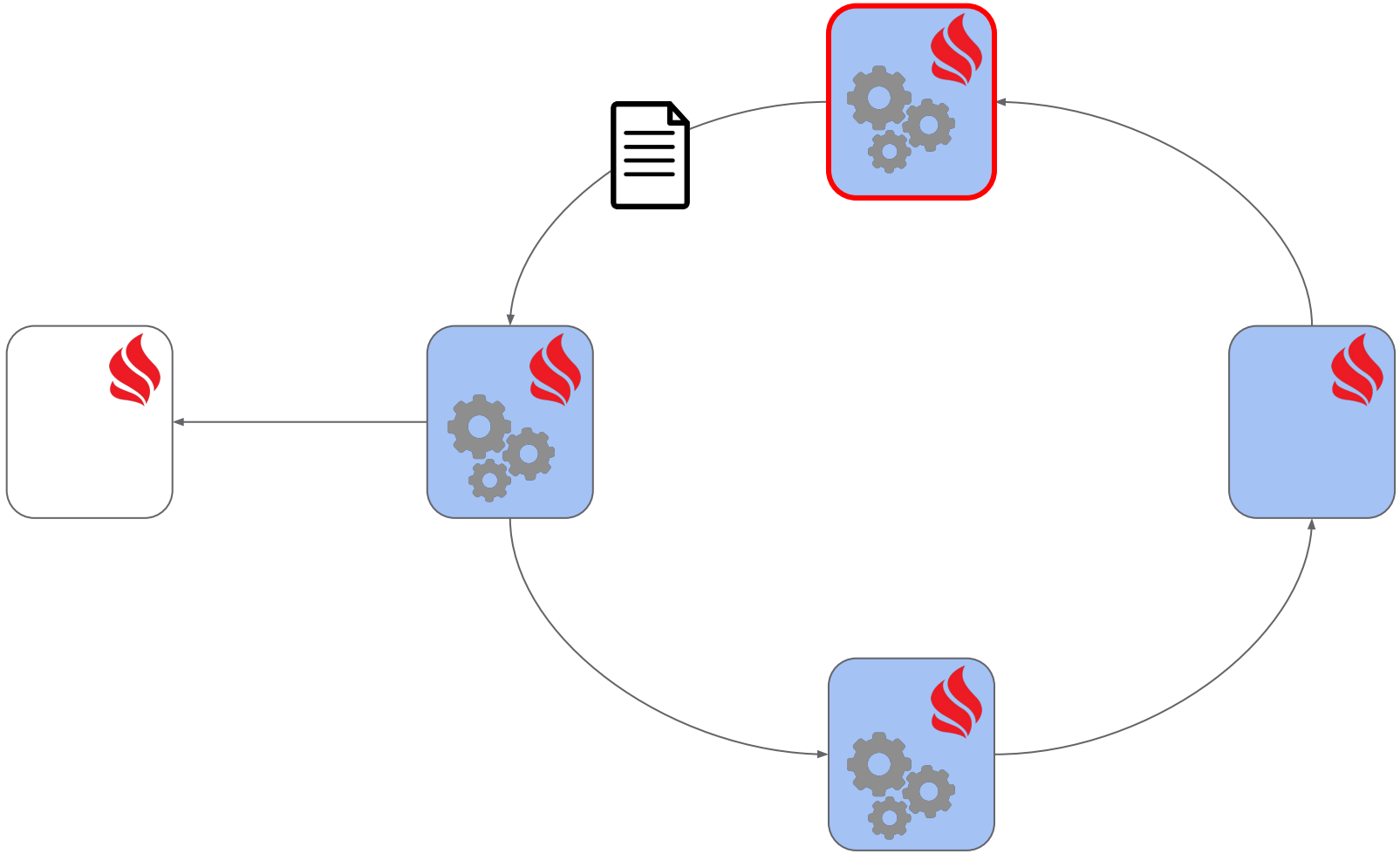
Deployment procedure (2/4)



Deployment procedure (3/4)

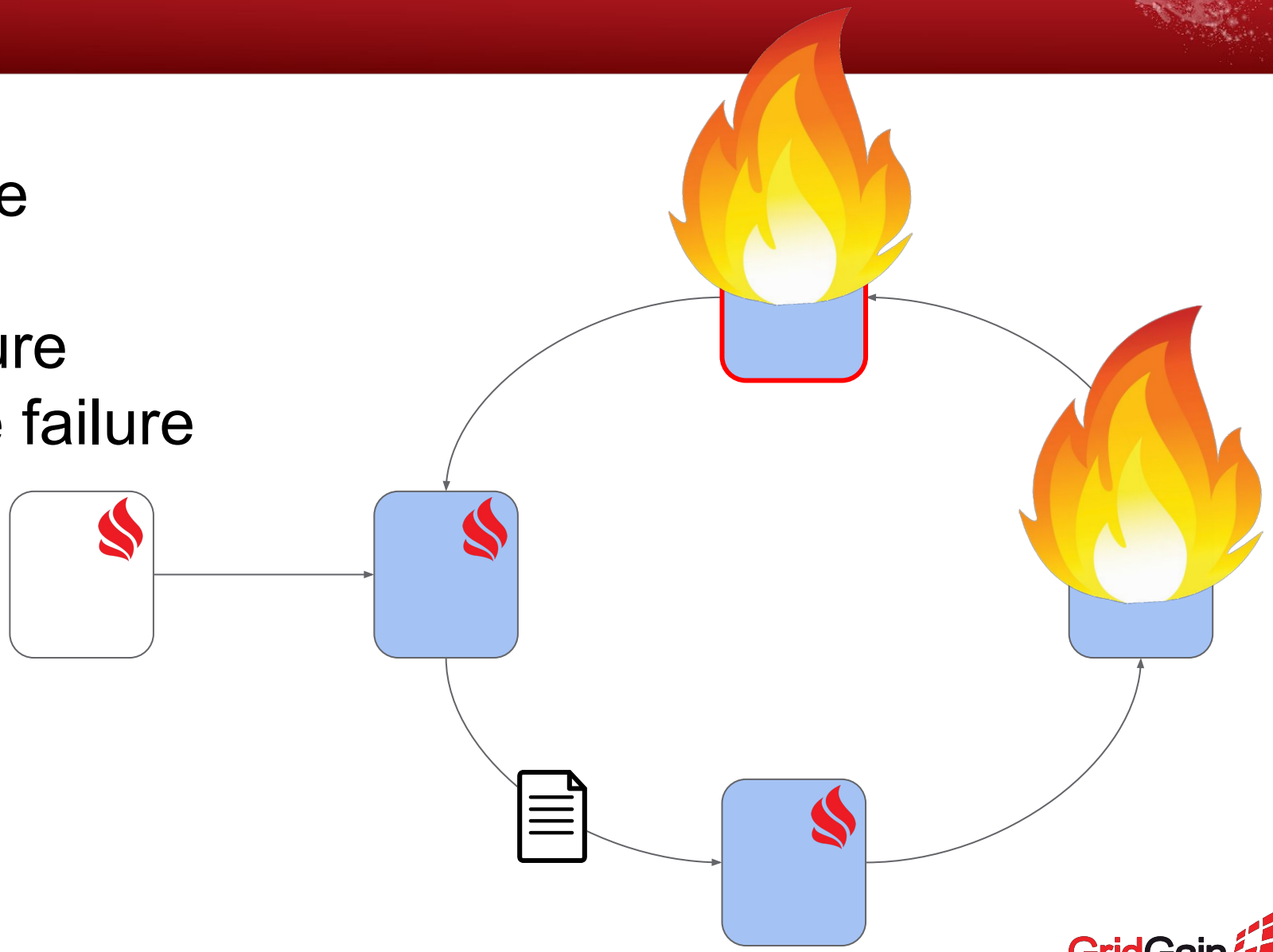


Deployment procedure (4/4)



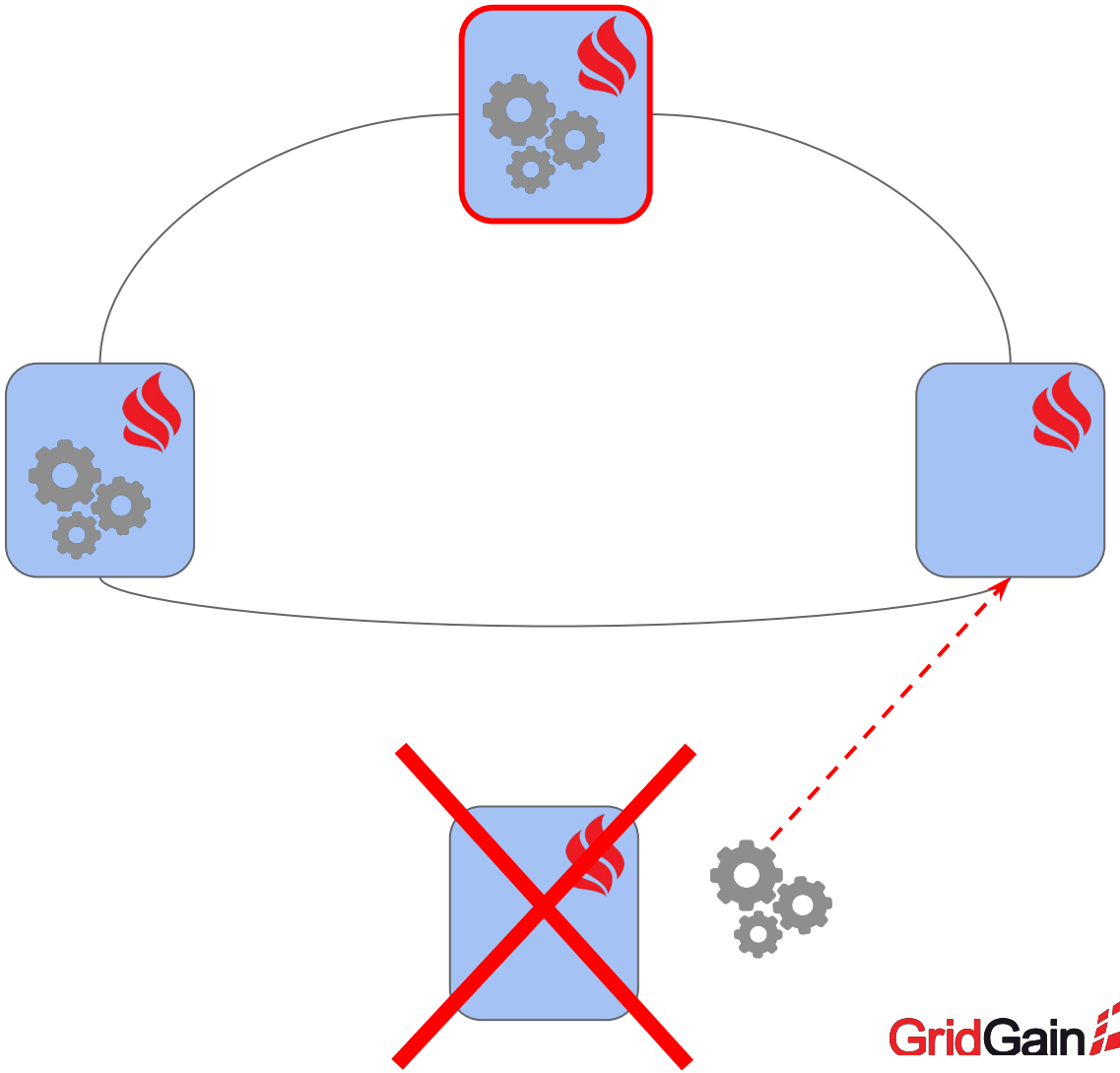
Failure handling

- Coordinator failure
- Client failure
- Service node failure
- Non-service node failure



Topology change

- Recalculate the distribution
- Repeat the deployment procedure
- The service is moved to another node



Links

- User documentation: https://apacheignite.readme.io/docs/service-grid
- Design: https://bit.ly/iep-services
- Dev list discussion: http://bit.ly/services-dev-list
- Examples:
  - https://github.com/apache/ignite/tree/master/examples
  - https://github.com/dmekhanikov/ignite-text-search

GridGain

Thanks!

dmekhanikov@gmail.com