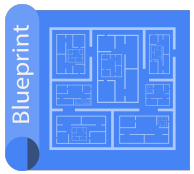
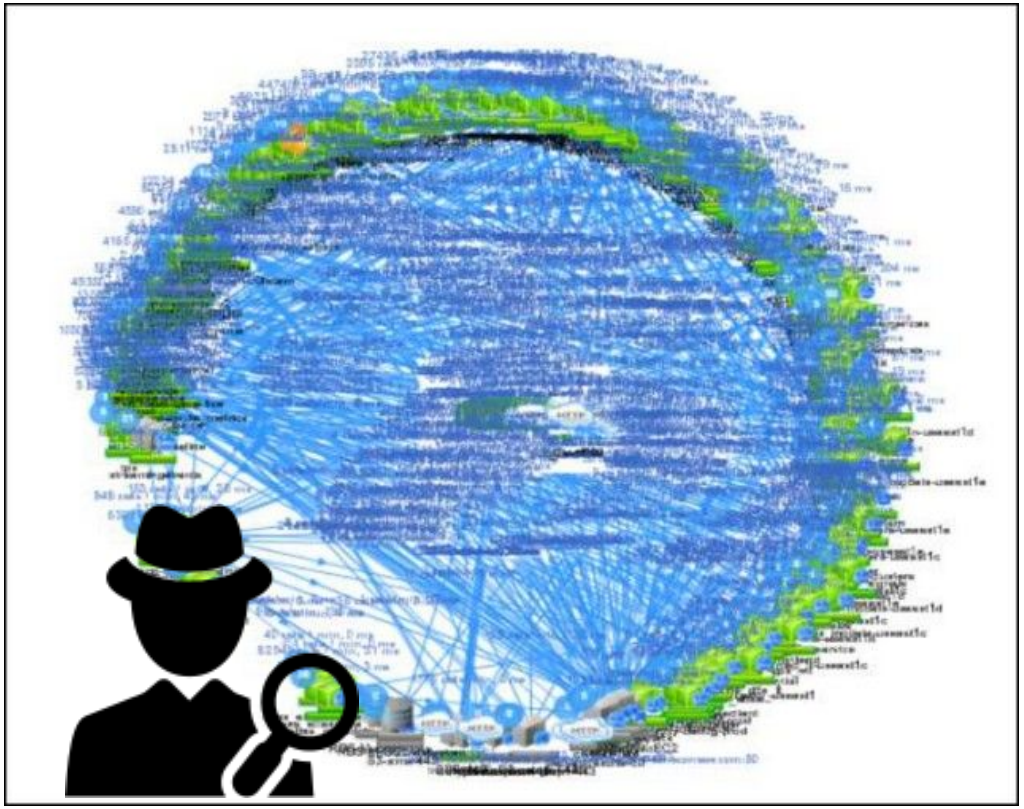


Blueprint: A Toolchain for Highly Reconfigurable Microservice Applications

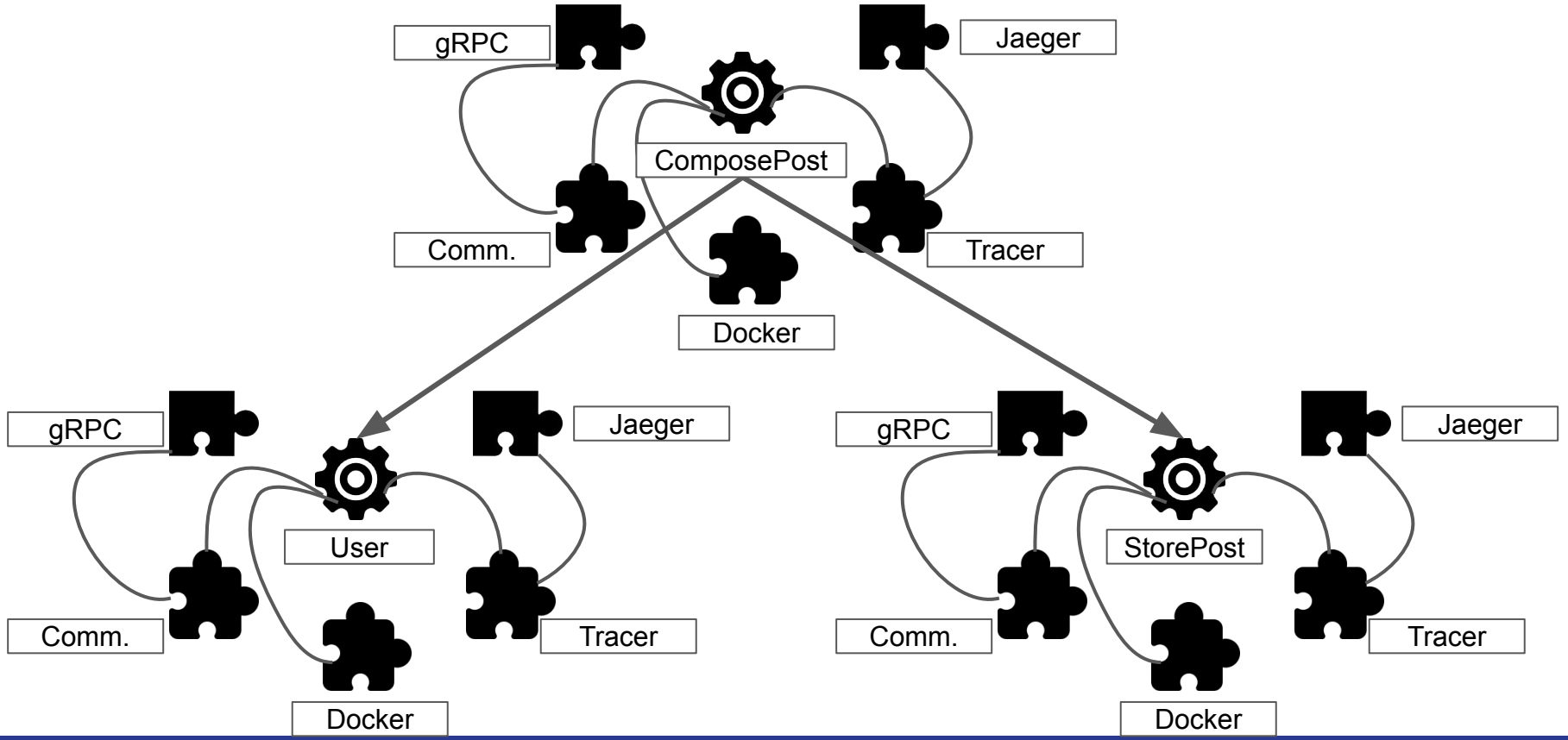
Vaastav Anand, Deepak Garg, Antoine Kaufmann, Jonathan Mace

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

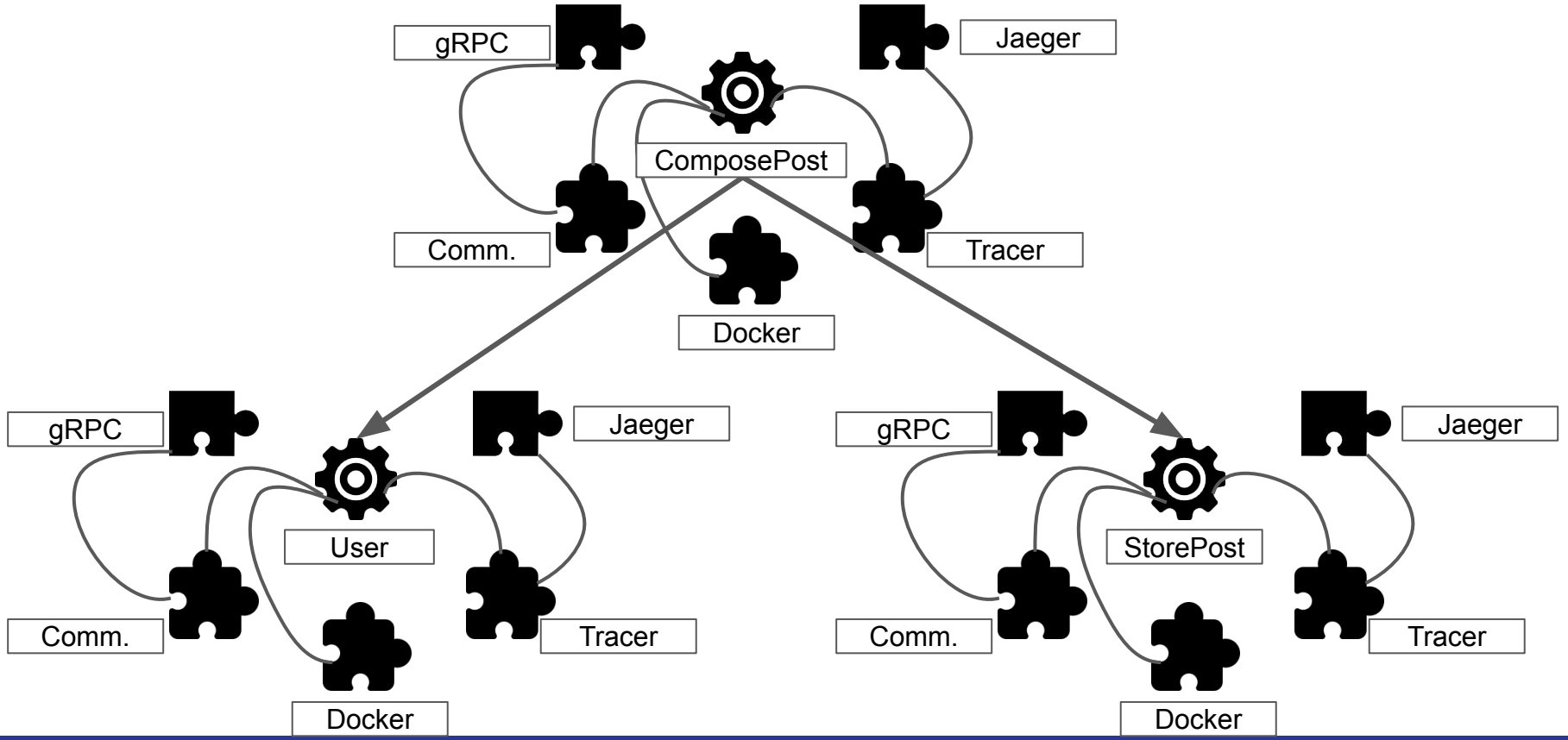
A Typical Microservice Application



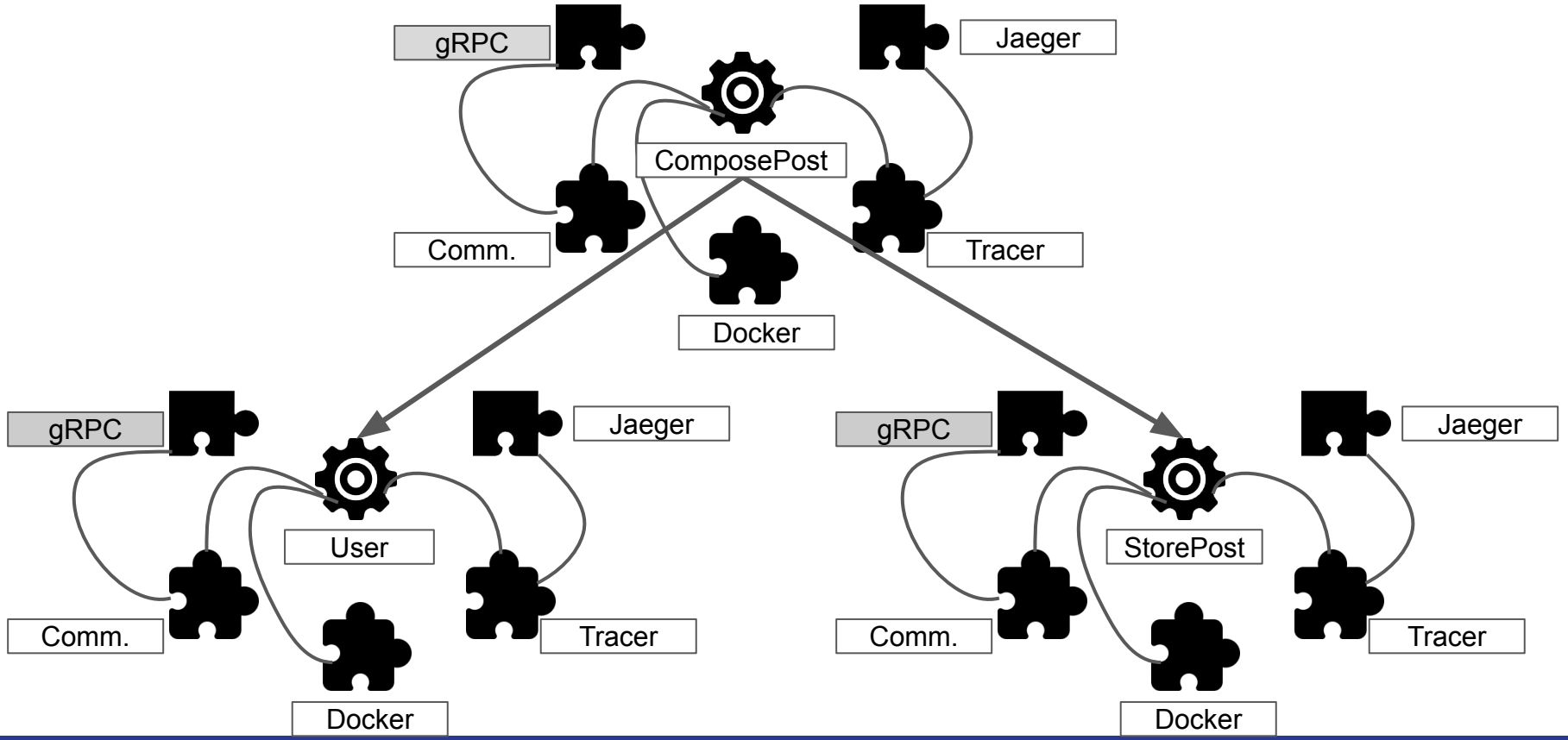
A Typical Microservice



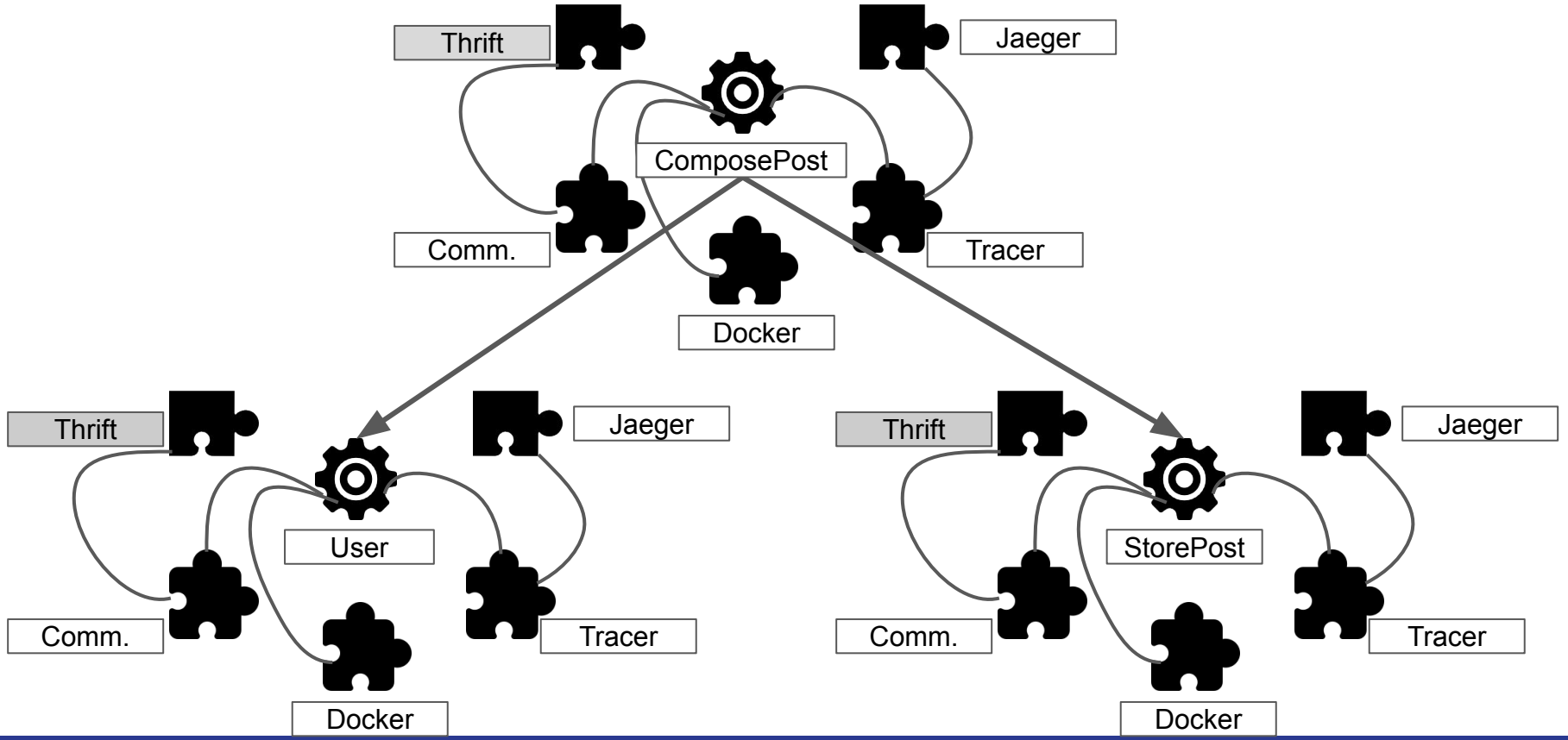
Microservice research requires modifications



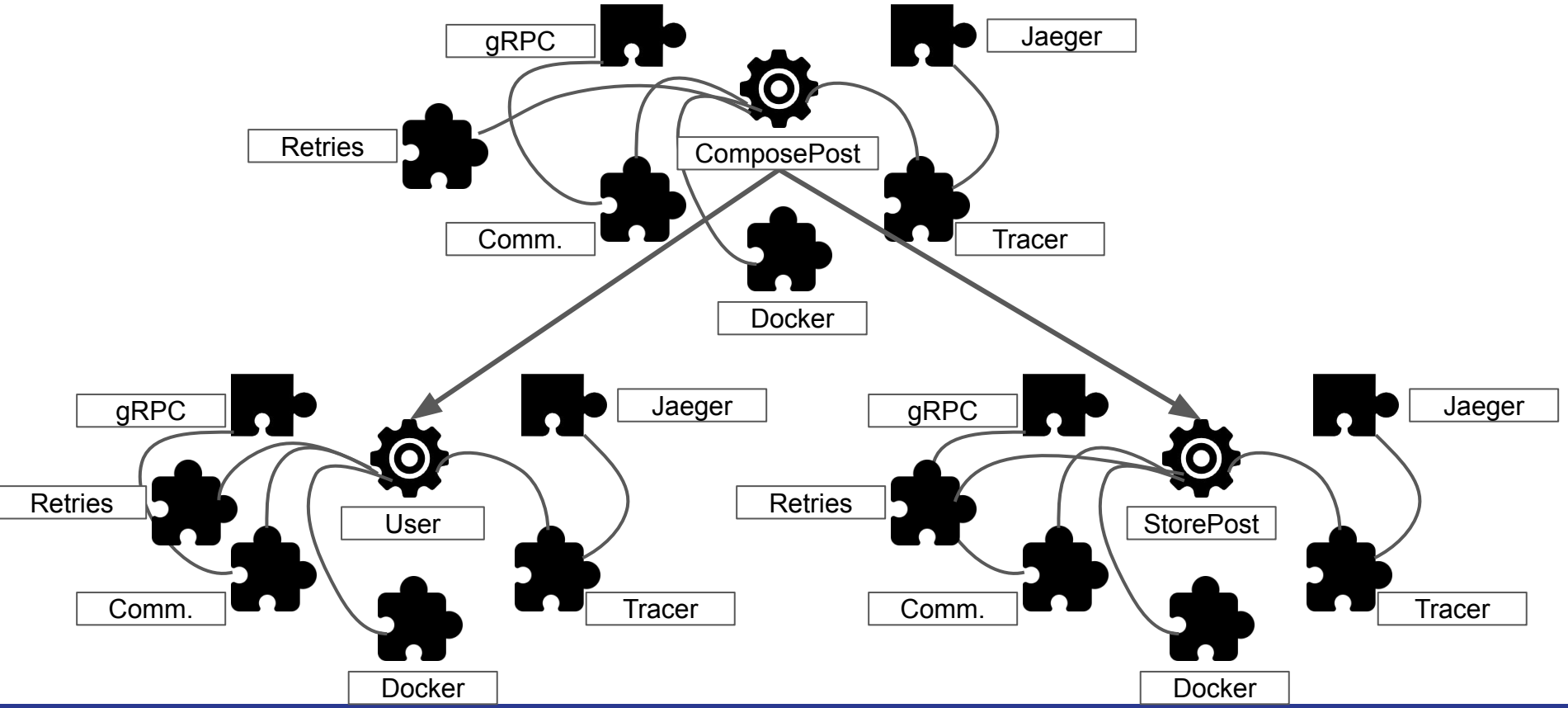
Microservice research requires modifications



Microservice research requires modifications



Microservice research requires modifications



Blueprint Overview

Blueprint: a **toolchain** for building **reconfigurable** microservice apps

- Change aspects of **the design of the system**
- **Generate** implementations corresponding to different designs

Blueprint Overview

Blueprint: a **toolchain** for building **reconfigurable** microservice apps

- Change aspects of **the design of the system**
- **Generate** implementations corresponding to different designs

Why do we need Blueprint?

A Typical Microservice is Inflexible!

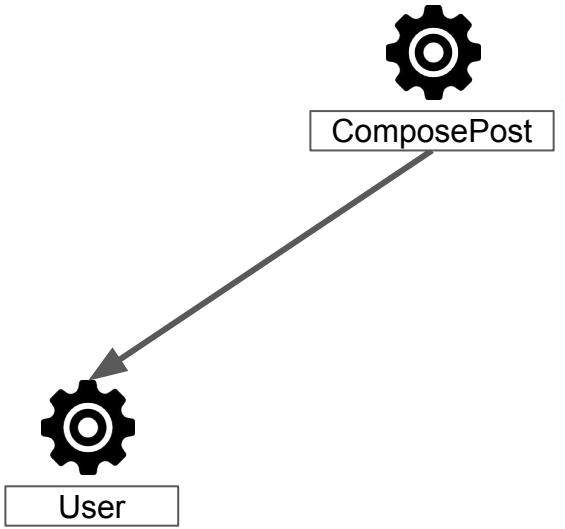

```
using json = nlohmann::json;
using std::chrono::duration_cast;
using std::chrono::milliseconds;
using std::chrono::system_clock;

class ComposePostHandler : public ComposePostServiceIf {
 public:
  ComposePostHandler(ClientPool<ThriftClient<PostStorageServiceClient>> *
  TextServiceReturn ComposePostHandler::_ComposeTextHelper(
      int64_t req_id, const std::string &text,
      const std::map<std::string, std::string> &carrier) {
    TextMapReader reader(carrier);
    auto parent_span = opentracing::Tracer::Global()->Extract(rea
    auto span = opentracing::Tracer::Global()->StartSpan(
        "compose_text_client", {opentracing::ChildOf(parent_span-
    std::map<std::string, std::string> writer_text_map;
    TextMapWriter writer(writer_text_map);
    opentracing::Tracer::Global()->Inject(span->context(), writer

    auto text_client_wrapper = _text_service_client_pool->Pop();
    if (!text_client_wrapper) {
      ServiceException se;
      se.errorCode = ErrorCode::SE_THRIFT_CONN_ERROR;
      se.message = "Failed to connect to text-service";
      LOG(error) << se.message;
      ;
      span->Finish();
      throw se;
    }

    auto text_client = text_client_wrapper->GetClient();
    TextServiceReturn _return_text;
    try {
      text_client->ComposeText(_return_text, req_id, text, writer
    } catch (...) {
      LOG(error) << "Failed to send compose-text to text-service"
      _text_service_client_pool->Remove(text_client_wrapper);
      span->Finish();
      throw;
    }
    _text_service_client_pool->Keepalive(text_client_wrapper);
    span->Finish();
    return _return_text;
  }
```

```
void _UploadUserTimelineHelper(
    int64_t req_id, int64_t post_id, int64_t user_id, int64_t ti
    const std::map<std::string,
std::vector<Media> ComposePostHandler::_ComposeMe
    int64_t req_id, const std::vector<std::string
    const std::vector<int64_t> &media_ids,
    const std::map<std::string, std::string> &car
  TextMapReader reader(carrier);
  auto parent_span = opentracing::Tracer::Global(
  auto span = opentracing::Tracer::Global()->Star
      "compose_media_client", {opentracing::Child
  std::map<std::string, std::string> writer_text_
  TextMapWriter writer(writer_text_map);
  opentracing::Tracer::Global()->Inject(span->con

  auto media_client_wrapper = _media_service_clie
  if (!media_client_wrapper) {
    ServiceException se;
    se.errorCode = ErrorCode::SE_THRIFT_CONN_ERRO
    se.message = "Failed to connect to media-serv
    LOG(error) << se.message;
    ;
    span->Finish();
    throw se;
  }

  auto media_client = media_client_wrapper->GetCl
  std::vector<Media> _return_media;
  try {
    media_client->ComposeMedia(_return_media, req
                               writer_text_map);
  } catch (...) {
    LOG(error) << "Failed to send compose-media t
    _media_service_client_pool->Remove(media_client_wrapper);
    span->Finish();
    throw;
  }
  _media_service_client_pool->Keepalive(media_client_wrapper);
  span->Finish();
  return _return_media;
}
```

```
Creator ComposePostHandler::_ComposeCreaterHelper(
int64_t Comp
    int64_t
    const st
TextMapRea
auto paren
auto span
    "compo
std::map<s
TextMapWri
opentracin

auto uniqu
if (!uniqu
  ServiceE
  se.error
  se.messa
  LOG(erro
  span->Fi
  throw se
}

auto uniqu
int64_t _r
try {
  _return_
      uniq
} catch (.
  LOG(erro
  _unique_
  span->Fi
  throw;
}
_unique_id
span->Fini
return _re
}
```

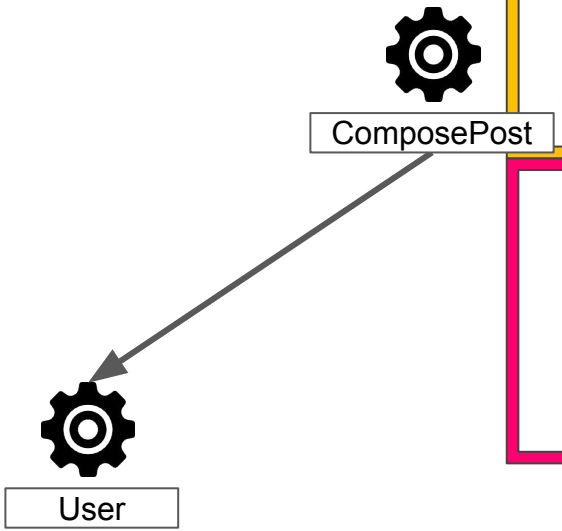
```
void ComposePostHandler::_UploadPostHelper(
    int64_t req_id, const Post &post,
    const std::map<std::string, std::string> &carrier) {
  TextMapReader reader(carrier);
  auto parent_span = opentracing::Tracer::Global()->Extract(reader);
  auto span = opentracing::Tracer::Global()->StartSpan(
      "store_post_client", {opentracing::ChildOf(parent_span->get())});
  std::map<std::string, std::string> writer_text_map;
  TextMapWriter writer(writer_text_map);
  opentracing::Tracer::Global()->Inject(span->context(), writer);

  auto post_storage_client_wrapper = _post_storage_client_pool->Pop();
  if (!post_storage_client_wrapper) {
    ServiceException se;
    se.errorCode = ErrorCode::SE_THRIFT_CONN_ERROR;
    se.message = "Failed to connect to post-storage-service";
    LOG(error) << se.message;
    ;
    throw se;
  }
  auto post_storage_client = post_storage_client_wrapper->GetClient();
  try {
    post_storage_client->StorePost(req_id, post, writer_text_map);
  } catch (...) {
    _post_storage_client_pool->Remove(post_storage_client_wrapper);
    LOG(error) << "Failed to store post to post-storage-service";
    throw;
  }
  _post_storage_client_pool->Keepalive(post_storage_client_wrapper);

  span->Finish();
}
```

Microservice Components are tightly coupled




```
Creator ComposePostHandler::_ComposeCreaterHelper(
    int64_t req_id, int64_t user_id, const std::string &username,
    const std::map<std::string, std::string> &carrier) {
  TextMapReader reader(carrier);
  auto parent_span = opentracing::Tracer::Global()->Extract(reader);
  auto span = opentracing::Tracer::Global()->StartSpan(
      "compose_creator_client", {opentracing::ChildOf(parent_span->get())});
  std::map<std::string, std::string> writer_text_map;
  TextMapWriter writer(writer_text_map);
  opentracing::Tracer::Global()->Inject(span->context(), writer);

  auto user_client_wrapper = _user_service_client_pool->Pop();
  if (!user_client_wrapper) {
    ServiceException se;
    se.errorCode = ErrorCode::SE_THRIFT_CONN_ERROR;
    se.message = "Failed to connect to user-service";
    LOG(error) << se.message;
    span->Finish();
    throw se;
  }

  auto user_client = user_client_wrapper->GetClient();
  Creator _return_creator;
  try {
    user_client->ComposeCreatorWithUserId(_return_creator, req_id, user_id,
                                          username, writer_text_map);
  } catch (...) {
    LOG(error) << "Failed to send compose-creator to user-service";
    _user_service_client_pool->Remove(user_client_wrapper);
    span->Finish();
    throw;
  }
  _user_service_client_pool->Keepalive(user_client_wrapper);
  span->Finish();
  return _return_creator;
}
```

Microservice Components are tightly coupled

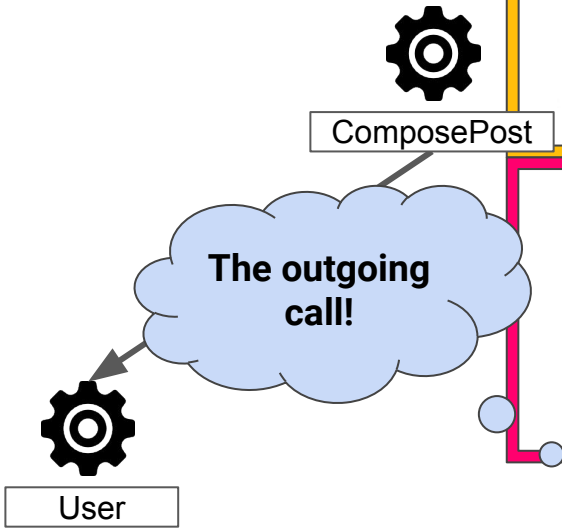



```
Creator ComposePostHandler::_ComposeCreaterHelper(
    int64_t req_id, int64_t user_id, const std::string &username,
    const std::map<std::string, std::string> &carrier) {
  TextMapReader reader(carrier);
  auto parent_span = opentracing::Tracer::Global()->Extract(reader);
  auto span = opentracing::Tracer::Global()->StartSpan(
      "compose_creator_client", {opentracing::ChildOf(parent_span->get())});
  std::map<std::string, std::string> writer_text_map;
  TextMapWriter writer(writer_text_map);
  opentracing::Tracer::Global()->Inject(span->context(), writer);

  auto user_client_wrapper = _user_service_client_pool->Pop();
  if (!user_client_wrapper) {
    ServiceException se;
    se.errorCode = ErrorCode::SE_THRIFT_CONN_ERROR;
    se.message = "Failed to connect to user-service";
    LOG(error) << se.message;
    span->Finish();
    throw se;
  }

  auto user_client = user_client_wrapper->GetClient();
  Creator _return_creator;
  try {
    user_client->ComposeCreatorWithUserId(_return_creator, req_id, user_id,
                                          username, writer_text_map);
  } catch (...) {
    LOG(error) << "Failed to send compose-creator to user-service";
    _user_service_client_pool->Remove(user_client_wrapper);
    span->Finish();
    throw;
  }
  _user_service_client_pool->Keepalive(user_client_wrapper);
  span->Finish();
  return _return_creator;
}
```

Tracing

Clientpools + Thrift

Microservice Components are tightly coupled




```
Creator ComposePostHandler::_ComposeCreaterHelper(
    int64_t req_id, int64_t user_id, const std::string &username,
    const std::map<std::string, std::string> &carrier) {
  TextMapReader reader(carrier);
  auto parent_span = opentracing::Tracer::Global()->Extract(reader);
  auto span = opentracing::Tracer::Global()->StartSpan(
      "compose_creator_client", {opentracing::ChildOf(parent_span->get())});
  std::map<std::string, std::string> writer_text_map;
  TextMapWriter writer(writer_text_map);
  opentracing::Tracer::Global()->Inject(span->context(), writer);

  auto user_client_wrapper = _user_service_client_pool->Pop();
  if (!user_client_wrapper) {
    ServiceException se;
    se.errorCode = ErrorCode::SE_THRIFT_CONN_ERROR;
    se.message = "Failed to connect to user-service";
    LOG(error) << se.message;
    span->Finish();
    throw se;
  }

  auto user_client = user_client_wrapper->GetClient();
  Creator _return_creator;
  try {
    user_client->ComposeCreatorWithUserId(_return_creator, req_id, user_id,
                                          username, writer_text_map);
  } catch (...) {
    LOG(error) << "Failed to send compose-creator to user-service";
    user_service_client_pool->Remove(user_client_wrapper);
    span->Finish();
    throw;
  }
  user_service_client_pool->Keepalive(user_client_wrapper);
  span->Finish();
  return _return_creator;
}
```

Tracing

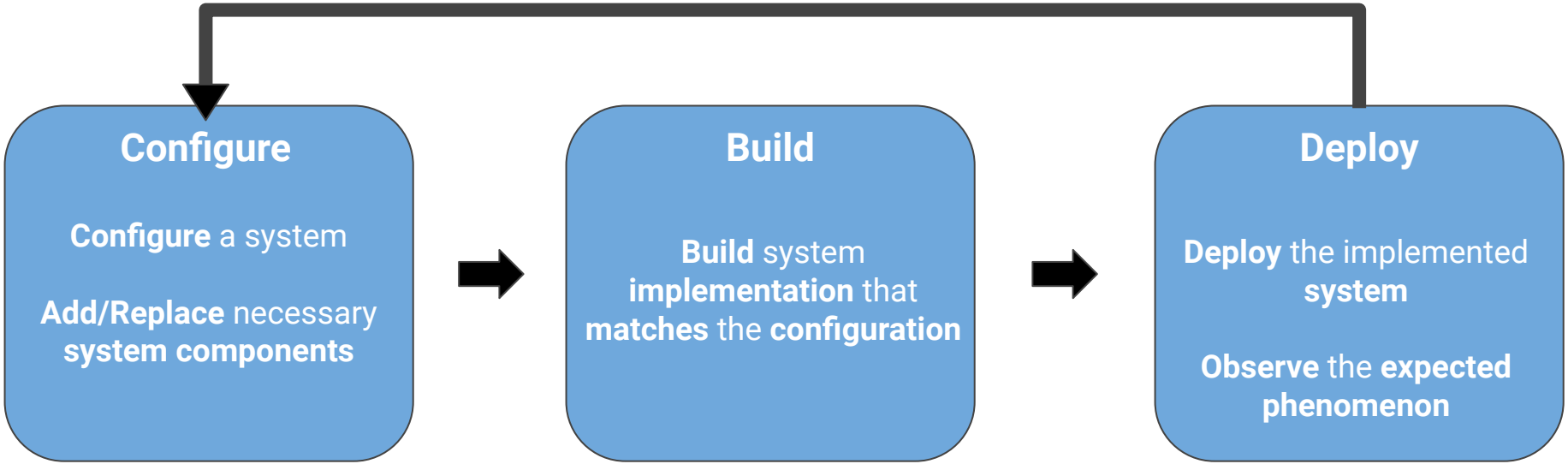
Clientpools + Thrift

Modifications are time-consuming

Modifying systems requires high technical **implementation effort**

- Changes can be **deep in the infrastructure**
  - eg: changing 1 service from thrift to grpc in DeathStarBench-socialnetwork: >1000 LoC
- Changes can be **cross-cutting**
  - eg: adding xtrace support to DeathStarBench-socialnetwork: 1289 LoC

Microservice research requires **C**onfigure, **B**uild, **D**eploy



Blueprint Contributions

Blueprint enables **Configure, Build, Deploy** as a first-class use-case

- **Set of abstractions** to simplify implementing and modifying apps
- **Novel compiler** to generate variant implementations

Blueprint Contributions

Blueprint enables **Configure, Build, Deploy** as a first-class use-case

- **Set of abstractions** to simplify implementing and modifying apps
- **Novel compiler** to generate variant implementations

Blueprint **reduces manual effort** for modifying systems

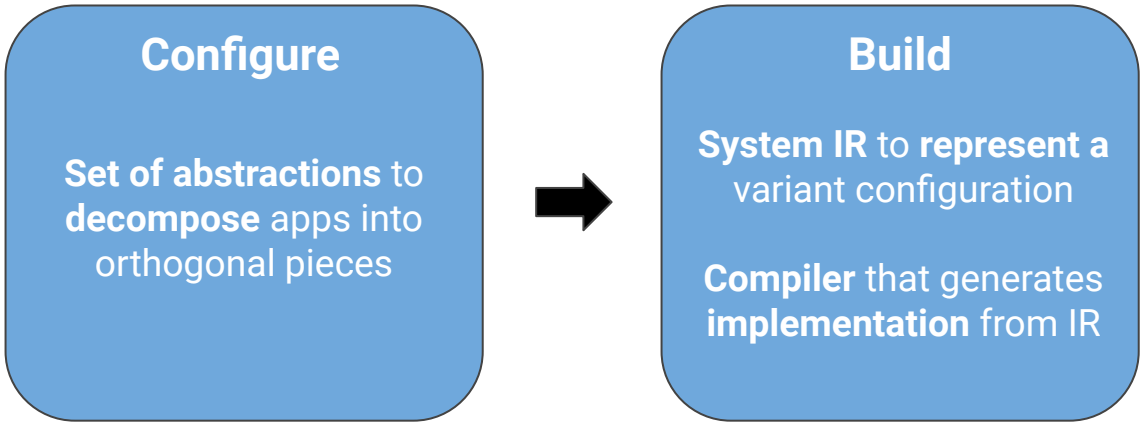
- **5-7x reduction** in implementation effort for developing applications
- Re-configuring systems takes few lines of code changes
  - eg: **1 line modified** to change the RPC framework

How does Blueprint enable Configure, Build, Deploy?

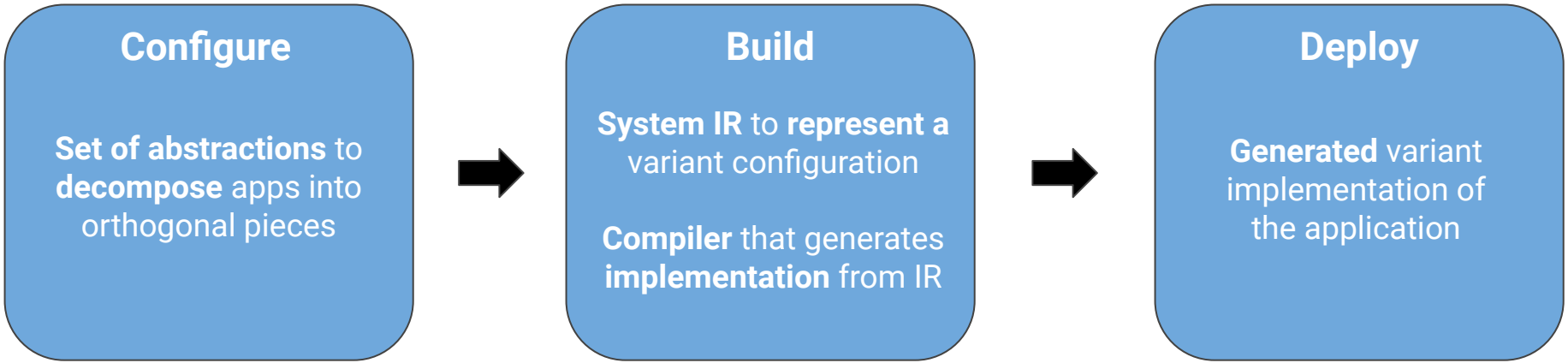
Configure

Set of abstractions to **decompose** apps into orthogonal pieces

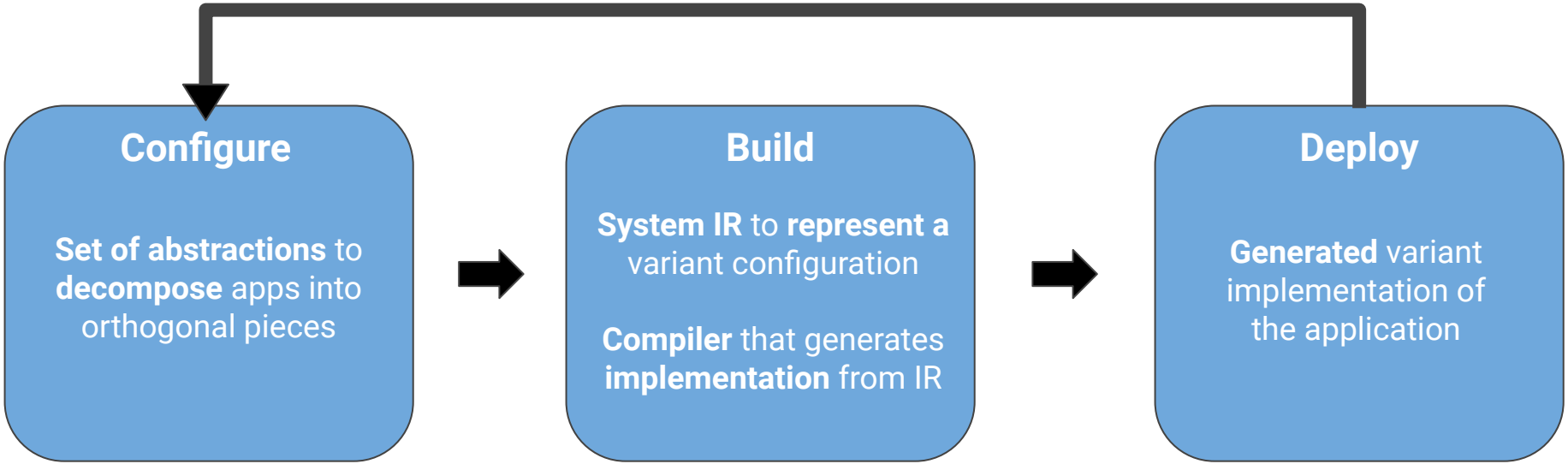
How does Blueprint enable Configure, Build, Deploy?



How does Blueprint enable Configure, Build, Deploy?

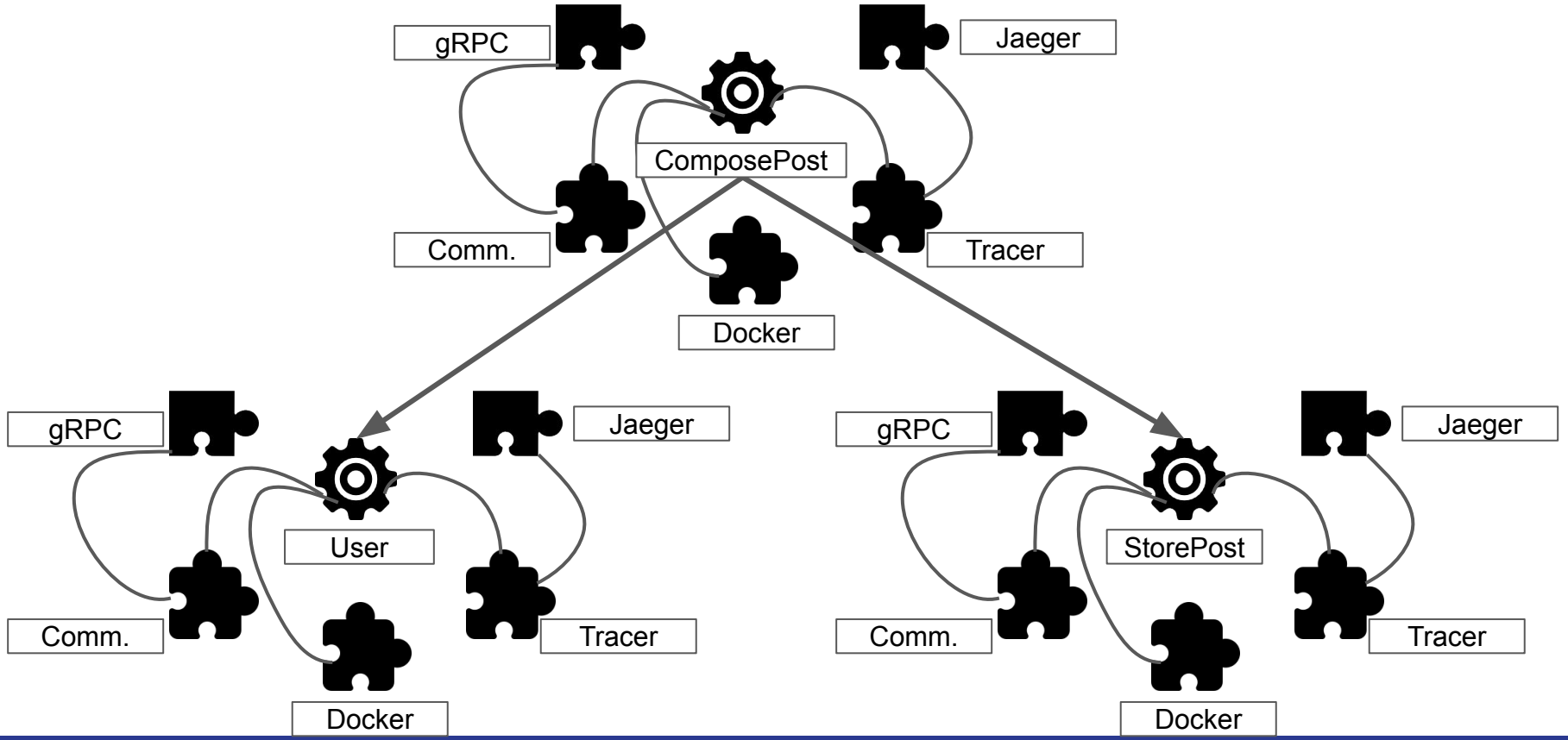


How does Blueprint enable Configure, Build, Deploy?

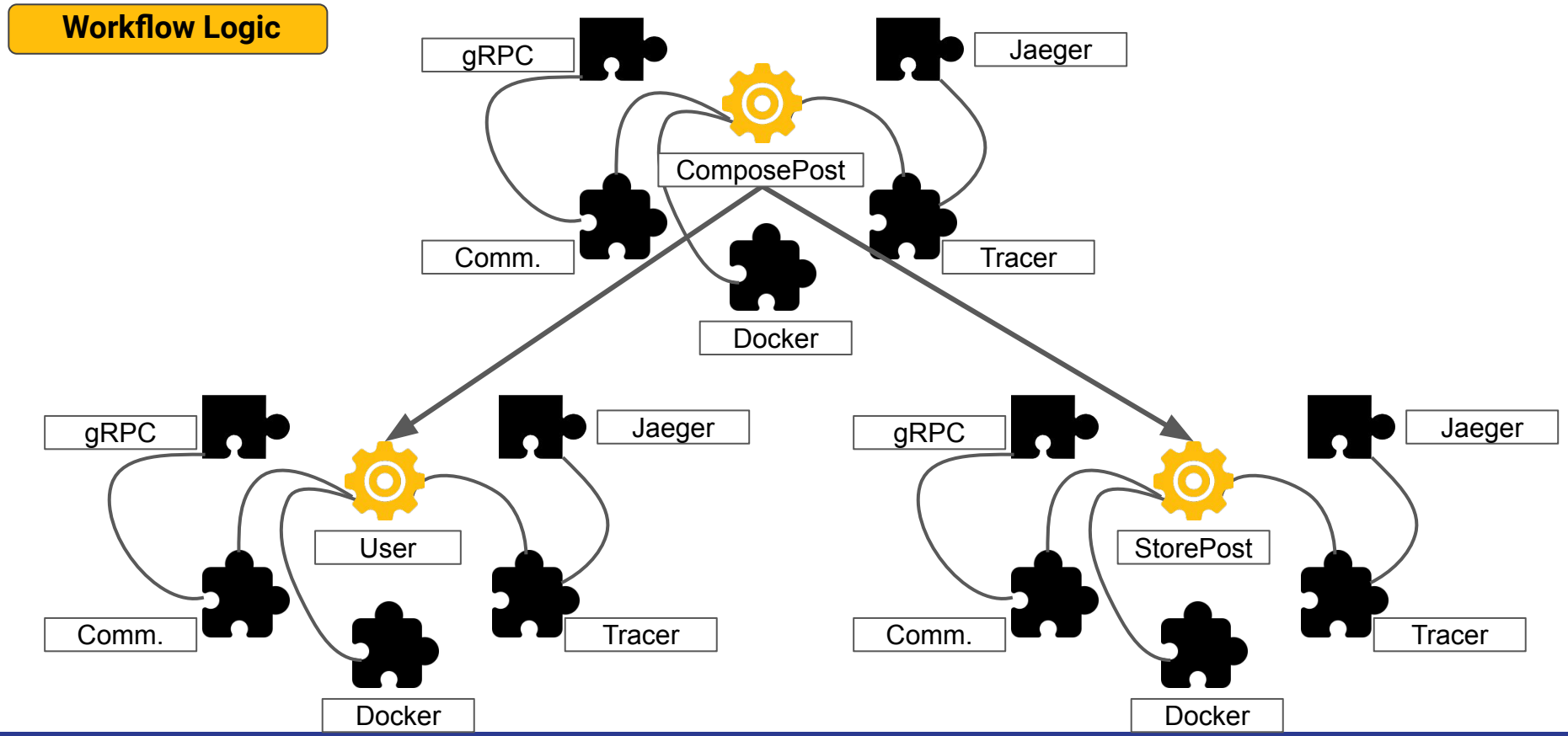


Key Idea: Decouple Application into three pieces

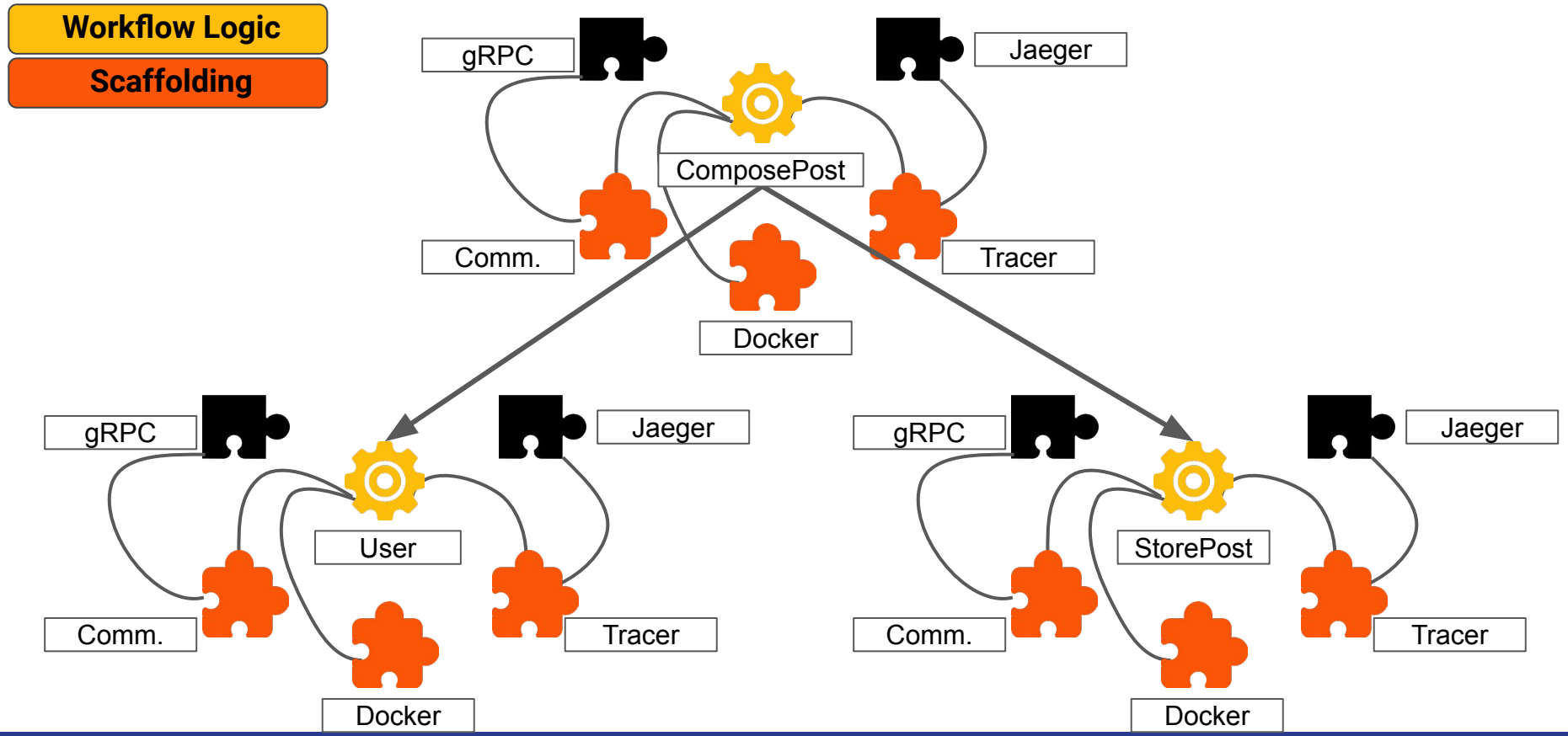
Key Idea: Decouple Application into three pieces



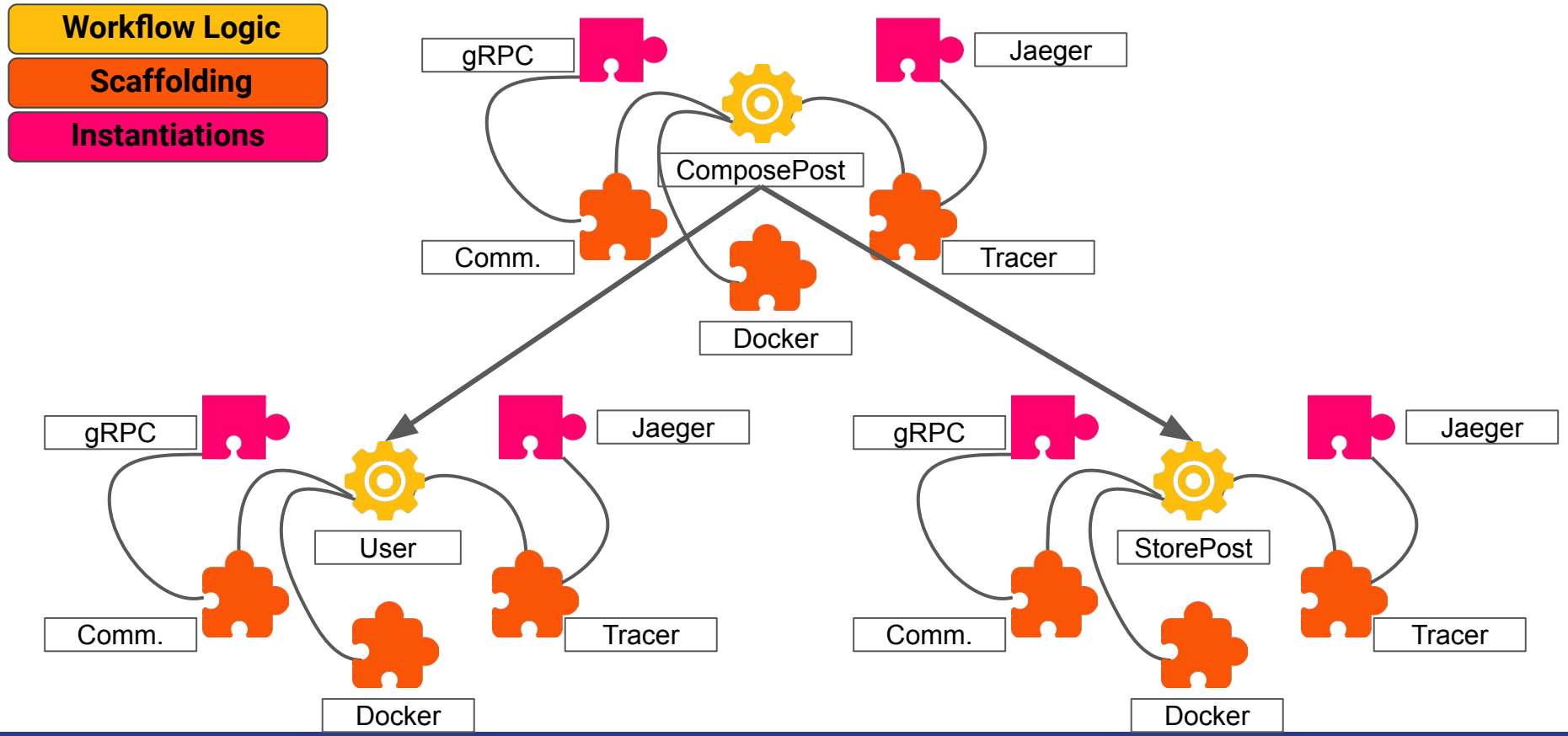
Key Idea: Decouple Application into three pieces



Key Idea: Decouple Application into three pieces



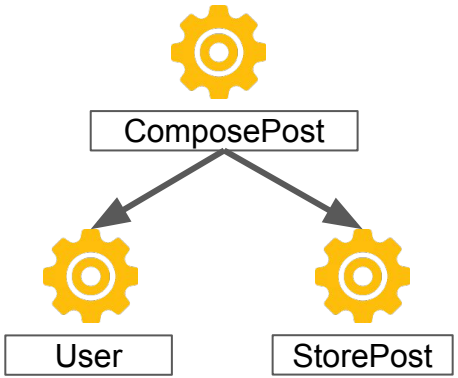
Key Idea: Decouple Application into three pieces



Separate 3 pieces into 2 input specs

Workflow Spec

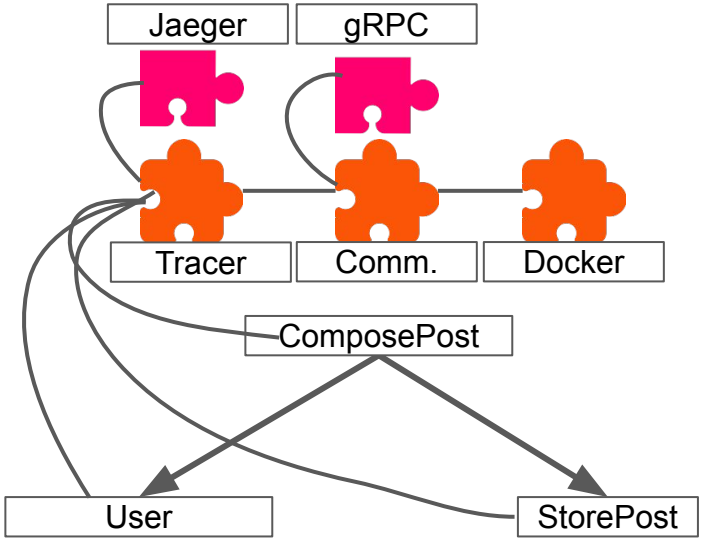
Workflow Logic



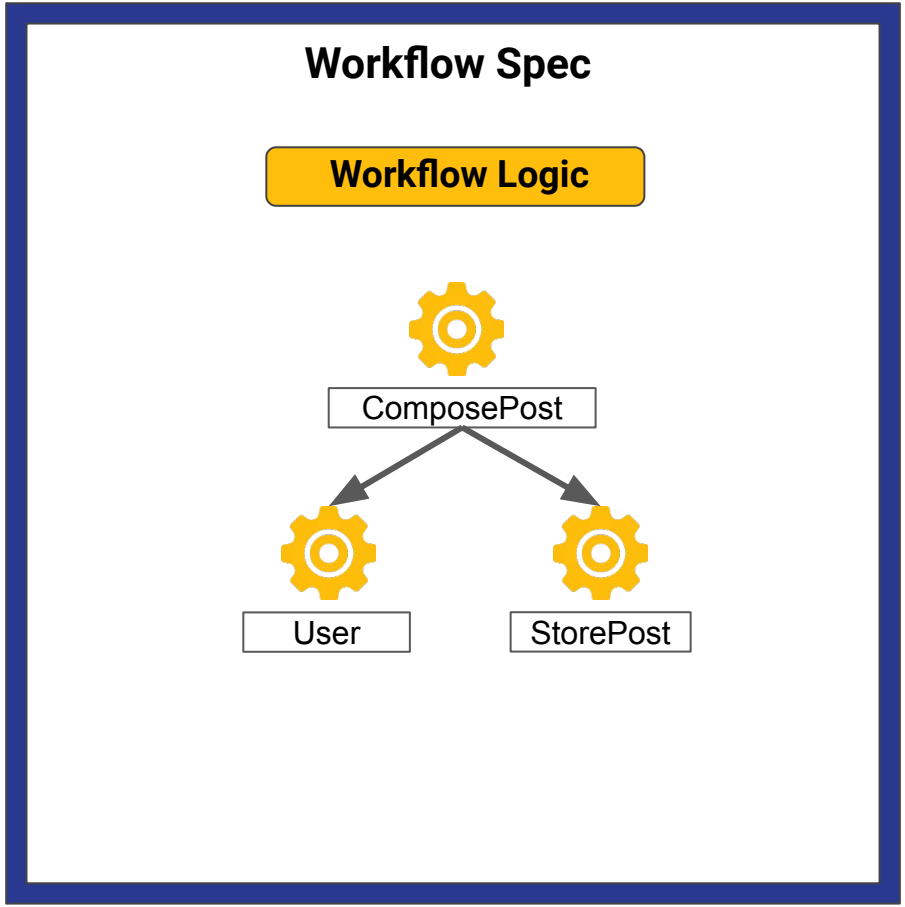
Wiring Spec

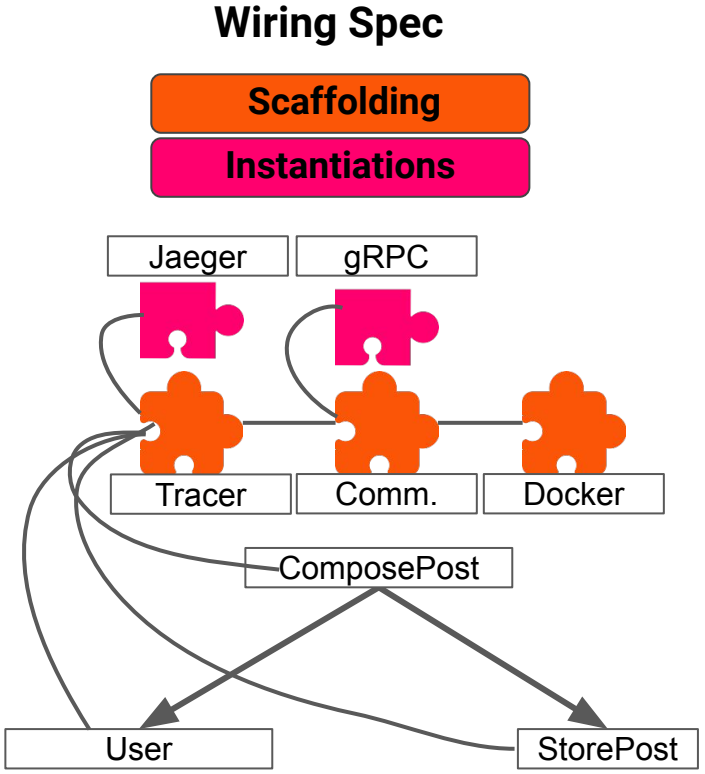
Scaffolding

Instantiations



Separate 3 pieces into 2 input specs





Workflow spec requires minimal effort

Workflow spec requires minimal effort

```
1  type ComposePostService interface {
2   ComposePost(userID int64, text postContent) error
3  }

4  type ComposePostImpl struct {
5   postStorageService PostStorageService
6   userService UserService
7  }

8  func NewComposePostImpl(ps PostStorageService, us UserService) *
         ComposePostService {
9   return &ComposePostImpl{ps, us}
10 }

11 func (c *ComposePostImpl) ComposePost(userID int64, text
         postContent) error {
12  creator, err := c.userService.GetUser(userId)
13  post := Post{Creator: creator, Text: text}
14  return c.postStorageService.StorePost(post)
15 }
```

Workflow spec requires minimal effort

Contains the service declarations

```
type ComposePostService interface {
 ComposePost(userID int64, text postContent) error
}

type ComposePostImpl struct {
 postStorageService PostStorageService
 userService UserService
}
```

```
       ComposePostService {
 9  return &ComposePostImpl{ps, us}
10  }

11  func (c *ComposePostImpl) ComposePost(userID int64, text
       postContent) error {
12   creator, err := c.userService.GetUser(userId)
13   post := Post{Creator: creator, Text: text}
14   return c.postStorageService.StorePost(post)
15  }
```

Workflow spec requires minimal effort

Contains the service declarations

```
type ComposePostService interface {
 ComposePost(userID int64, text postContent) error
}

type ComposePostImpl struct {
 postStorageService PostStorageService
 userService UserService
}

func NewComposePostImpl(ps PostStorageService, us UserService) *
      ComposePostService {
 return &ComposePostImpl{ps, us}
}

func (c *ComposePostImpl) ComposePost(userID int64, text
      postContent) error {
 creator, err := c.userService.GetUser(userId)
 post := Post{Creator: creator, Text: text}
 return c.postStorageService.StorePost(post)
}
```

Dependencies are passed as parameters to the constructor

Workflow spec requires minimal effort

Contains the service declarations

```
type ComposePostService interface {
 ComposePost(userID int64, text postContent) error
}

type ComposePostImpl struct {
 postStorageService PostStorageService
 userService UserService
}

func NewComposePostImpl(ps PostStorageService, us UserService) *
      ComposePostService {
 return &ComposePostImpl{ps, us}
}
```

Dependencies are passed as parameters to the constructor

Each function can be implemented in a few lines

```
func (c *ComposePostImpl) ComposePost(userID int64, text
      postContent) error {
 creator, err := c.userService.GetUser(userId)
 post := Post{Creator: creator, Text: text}
 return c.postStorageService.StorePost(post)
}
```

Workflow spec requires minimal effort

Contains the service declarations

```
type ComposePostService interface {
 ComposePost(userID int64, text postContent) error
}

type ComposePostImpl struct {
 postStorageService PostSto
 userService UserService
}

func NewComposePostImpl
       ComposePostServic
 return &ComposePostImpl{ps
}
```

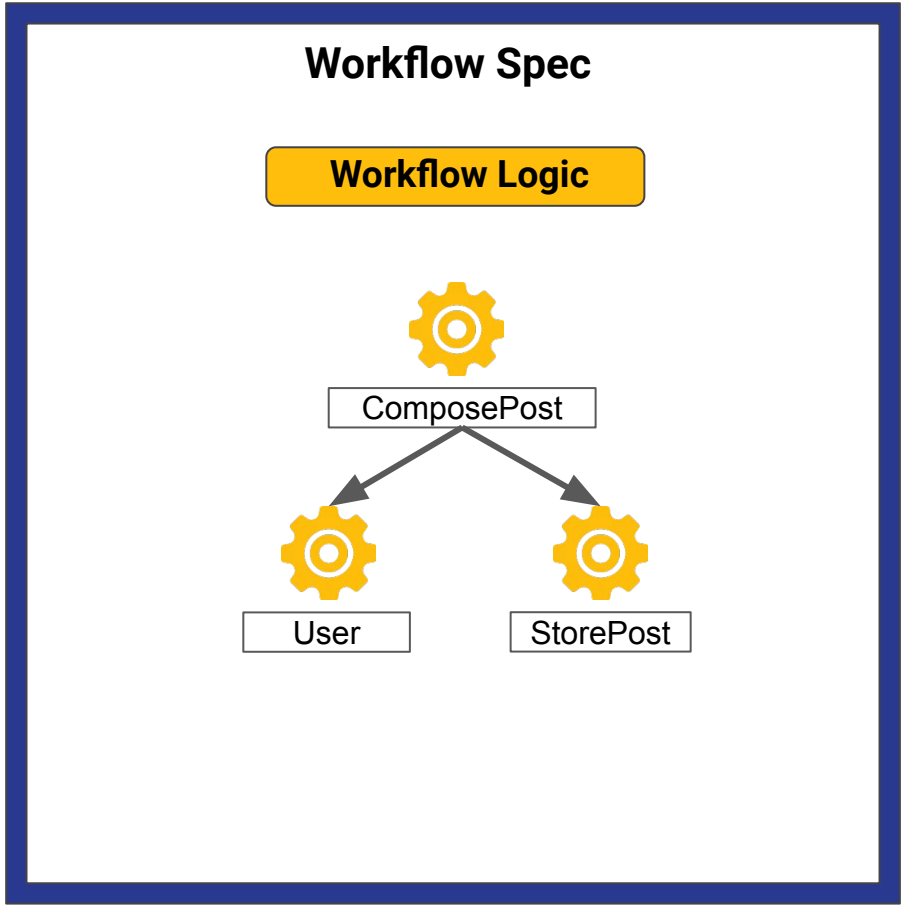
The outgoing call!

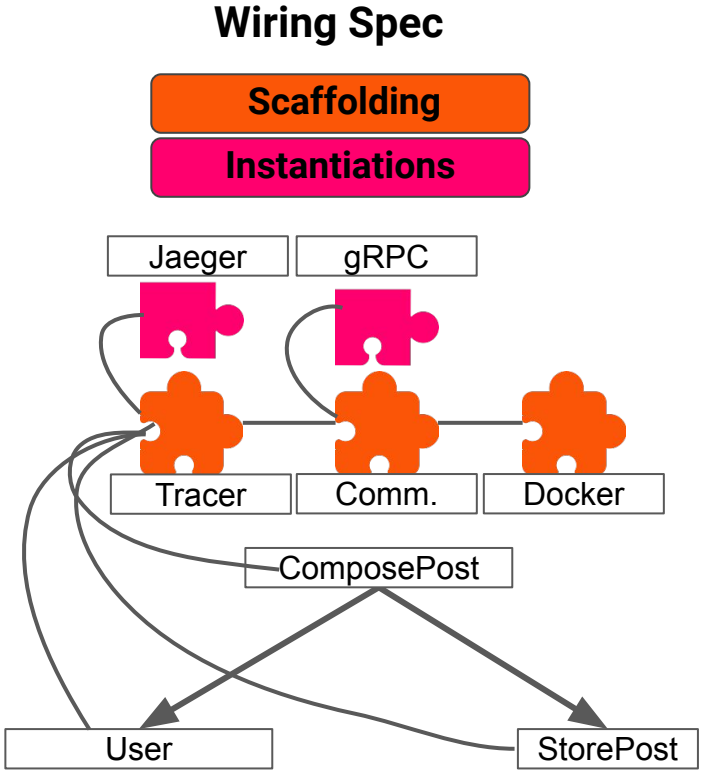
Dependencies are passed as parameters to the constructor

Each function can be implemented in a few lines

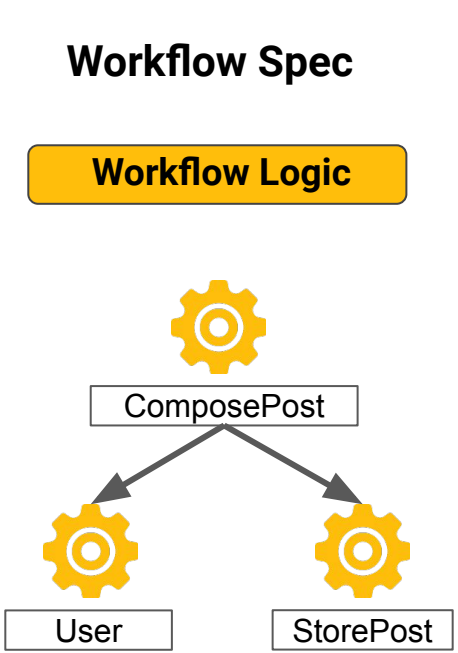
```
func (c *ComposePostImpl) ComposePost(userID int64, text
       postContent) error {
 creator, err := c.userService.GetUser(userId)
 post := Post{Creator: creator, Text: text}
 return c.postStorageService.StorePost(post)
}
```

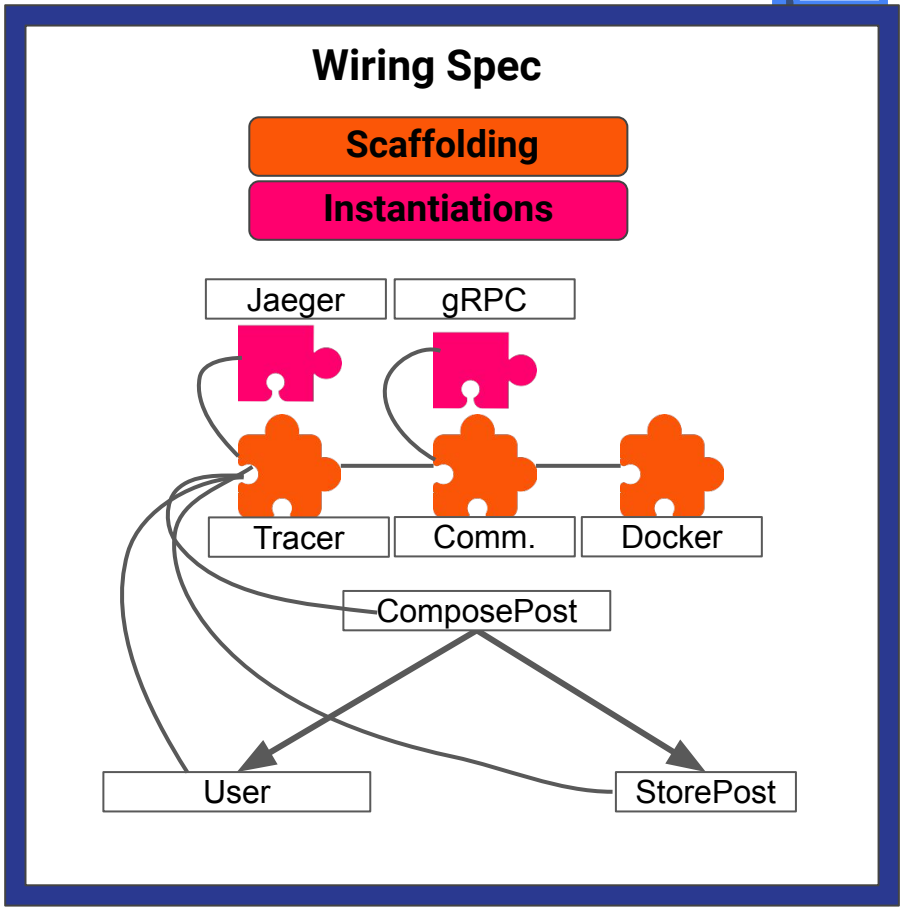
Separate 3 pieces into 2 input specs



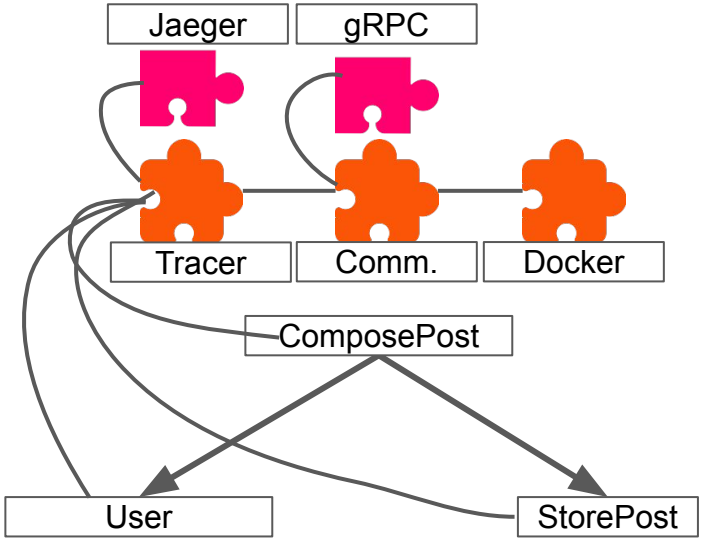


Separate 3 pieces into 2 input specs



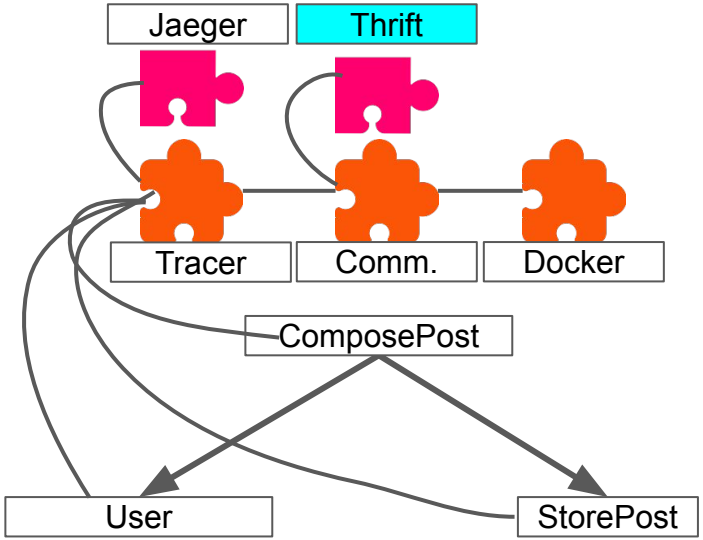


Wiring Spec specifies scaffolding and choices




```
normal_deployer : Modifier = Docker()
rpc_server : Modifier = GRPCServer()
tracer : Tracer = JaegerTracer()
tracerModifier: Modifier = TracerModifier(tracer=tracer)
server_modifiers : List[Modifier] = [rpc_server, normal_deployer,
      tracerModifier]
post_cache := Memcached()
post_db = MongoDB()
user_db = MongoDB()
us = UserServiceImpl(user_db).WithServer(server_modifiers)
ps = PostStorageServiceImpl(post_cache,post_db).WithServer(
      server_modifiers)
c1 = Container(ps,post_cache)
cs = ComposePostServiceImpl(ps, us).WithServer(server_modifiers)
```

Wiring Spec specifies scaffolding and choices




```
1  normal_deployer : Modifier = Docker()
2  rpc_server : Modifier = ThriftServer()
3  tracer : Tracer = JaegerTracer()
4  tracerModifier: Modifier = TracerModifier(tracer=tracer)
5  server_modifiers : List[Modifier] = [rpc_server, normal_deployer,
       tracerModifier]
6  post_cache := Memcached()
7  post_db = MongoDB()
8  user_db = MongoDB()
9  us = UserServiceImpl(user_db).WithServer(server_modifiers)
10 ps = PostStorageServiceImpl(post_cache,post_db).WithServer(
       server_modifiers)
11 c1 = Container(ps,post_cache)
12 cs = ComposePostServiceImpl(ps, us).WithServer(server_modifiers)
```

Change only requires modifications to few lines

Blueprint Compiler

Input: Workflow Spec + Wiring Spec

Output: Implementation of the system

Blueprint Compiler

Input: Workflow Spec + Wiring Spec

Output: Implementation of the system

Compilation Procedure:

- Convert input specs to graphical Intermediate Representation (IR)
  - Edges represent dependencies
  - Nodes represent instances, scaffolding, namespaces
- Generate implementation from IR using various plugins
  - Plugins are responsible for generating artifacts (code + config)

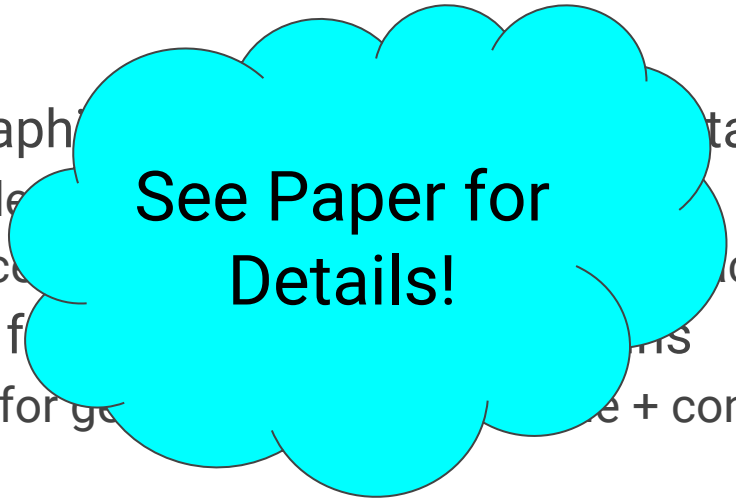
Blueprint Compiler

Input: Workflow Spec + Wiring Spec

Output: Implementation of the system

Compilation Procedure:

- Convert input specs to graph… …tation (IR)
  - Edges represent depende…
  - Nodes represent instanc… …ces
- Generate implementation f… …ns
  - Plugins are responsible for g… …e + config)



Implementation

All features and components implemented in 10K lines of code

Implementation

All features and components implemented in 10K lines of code

- Core compiler abstractions in 4K lines of Go code
- Wiring DSL is a Python-based DSL in 771 LOC.

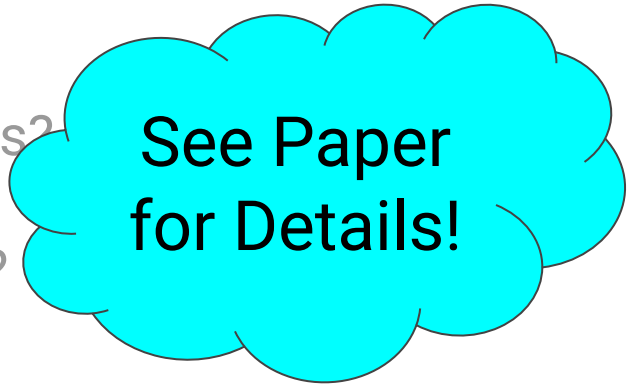
Implemented in Go and Python-based DSL

Evaluation Questions

- Do Blueprint's abstractions reduce programmer effort?
- Can Blueprint aid microservices research?
- Are Blueprint-generated systems realistic?
- Is Blueprint easy to extend with new features?
- What is the cost of Blueprint's abstractions?

Evaluation Questions

- **Do Blueprint's abstractions reduce programmer effort?**
- **Can Blueprint aid microservices research?**
- Are Blueprint-generated systems realistic?
- Is Blueprint easy to extend with new features?
- What is the cost of Blueprint's abstractions?



Evaluation Questions

- **Do Blueprint's abstractions reduce programmer effort?**
- Can Blueprint help aid in microservices research?

Blueprint reduces implementation effort

System Name	Original (LoC)	Workflow Spec (LoC)	Wiring Spec (LoC)	Reduction
DSB-SN	8209			
DSB-HR	5160			
DSB-MM	7794			
TrainTicket	54466			
SockShop	13987			

Blueprint reduces implementation effort

System Name	Original (LoC)	Workflow Spec (LoC)	Wiring Spec (LoC)	Reduction
DSB-SN	8209	1478	57	
DSB-HR	5160	679	63	
DSB-MM	7794	1401	42	
TrainTicket	54466	9639	166	
SockShop	13987	2261	40	

Blueprint reduces implementation effort

System Name	Original (LoC)	Workflow Spec (LoC)	Wiring Spec (LoC)	Reduction
DSB-SN	8209	1478	57	**5.4x**
DSB-HR	5160	679	63	**7.0x**
DSB-MM	7794	1401	42	**5.4x**
TrainTicket	54466	9639	166	**5.6x**
SockShop	13987	2261	40	**6.1x**

Blueprint reduces modification effort

Change software design **choices** for **improving performance**

Blueprint reduces modification effort

Change software design **choices** for **improving performance**

We modify **HotelReservation** from DeathStarBench

Blueprint reduces modification effort

Change software design **choices** for **improving performance**

We modify **HotelReservation** from DeathStarBench

Change Description	LoC modified
Change RPC framework (grpc->thrift)	1

Blueprint reduces modification effort

Change software design **choices** for **improving performance**

We modify **HotelReservation** from DeathStarBench

Change Description	LoC modified
Change RPC framework (grpc->thrift)	1
Change ClientPoolSize (512->4096)	1

Blueprint reduces modification effort

Change software design **choices** for **improving performance**

We modify **HotelReservation** from DeathStarBench

Change Description	LoC modified
Change RPC framework (grpc->thrift)	1
Change ClientPoolSize (512->4096)	1
Change Service Granularity (Microservice -> Monolith)	9

Evaluation Questions

- Do Blueprint's abstractions reduce programmer effort?
- **Can Blueprint help aid in microservices research?**

Can Blueprint aid in researching emergent phenomena?

Elicit phenomena on Blueprint generated **DSB-HR** and **DSB-SN**

- Generate different **variants** of systems
- Deploy variants with different **workloads**

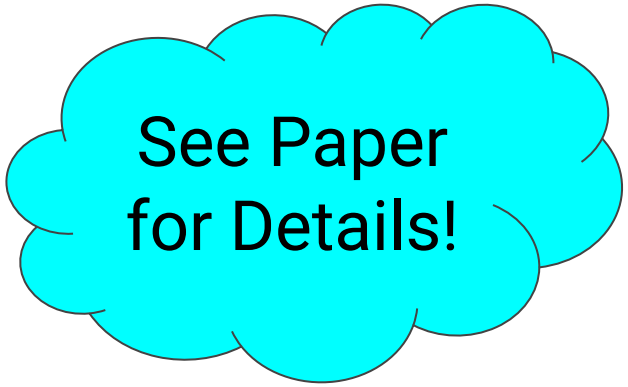
Can Blueprint aid in researching emergent phenomena?

Elicit phenomena on Blueprint generated **DSB-HR** and **DSB-SN**

- Generate different **variants** of systems
- Deploy variants with different **workloads**

Showcase two different **phenomena**

- Metastability failures (all 4 types)
- Cross-System Inconsistency



Reproducing Type 1 Metastability Failure

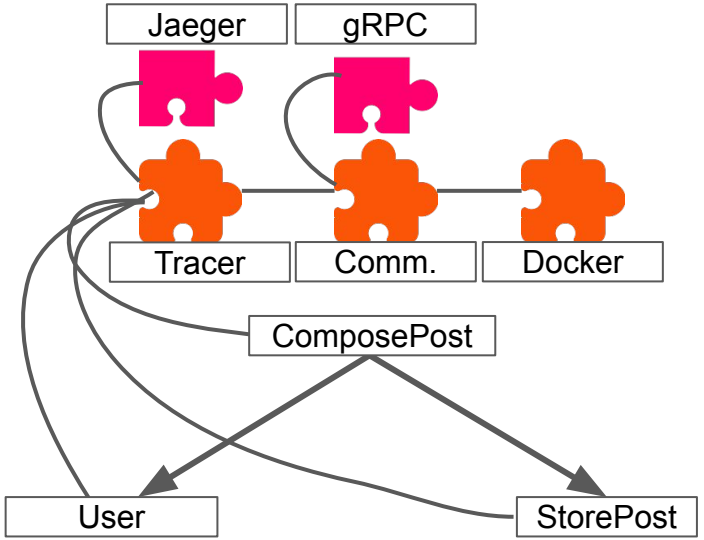
Type 1 Metastability Failure

- **Cause**
  - Increased Load -> More Request Timeouts
  - More Timeouts -> More Retries
  - More Retries -> Load Persists
  - Performance stays degraded!

Reproducing Type 1 Metastability Failure

Type 1 Metastability Failure

- **Cause**
  - Increased Load -> More Request Timeouts
  - More Timeouts -> More Retries
  - More Retries -> Load Persists
  - Performance stays degraded!

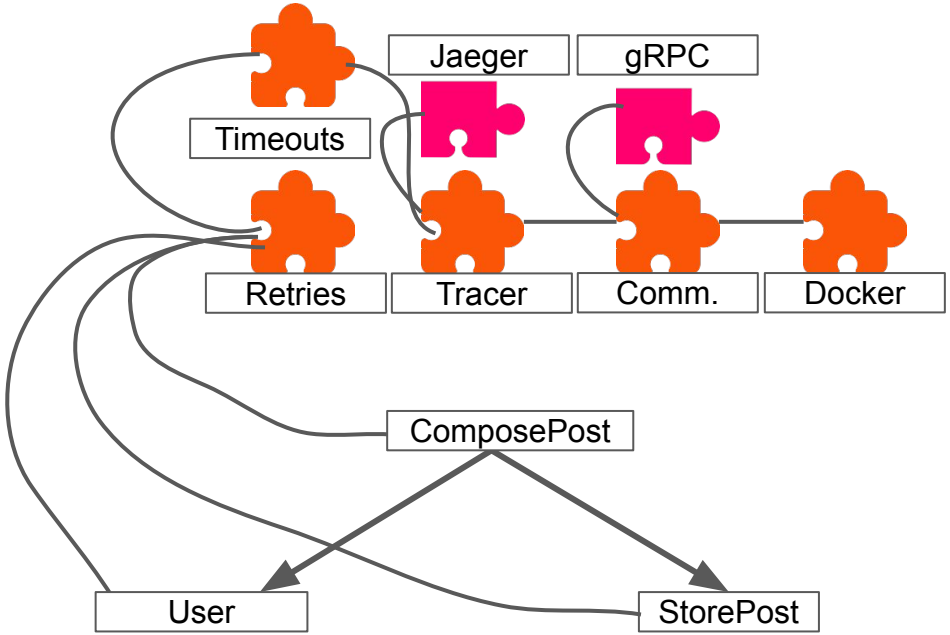


Reproducing Type 1 Metastability Failure

Type 1 Metastability Failure

- Cause
  - Increased Load -> More Request Timeouts
  - More Timeouts -> More Retries
  - More Retries -> Load Persists
  - Performance stays degraded!

Requires 2 lines of code changes!

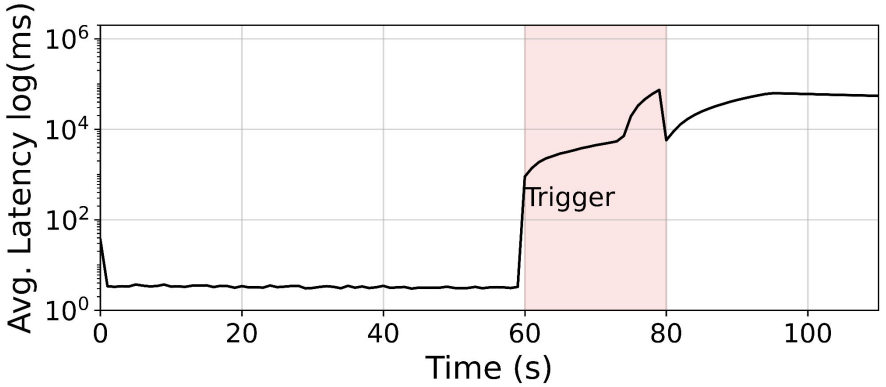


Reproducing Type 1 Metastability Failure

Type 1 Metastability Failure

- Cause
  - Increased Load -> More Request Timeouts
  - More Timeouts -> More Retries
  - More Retries -> Load Persists
  - Performance stays degraded!

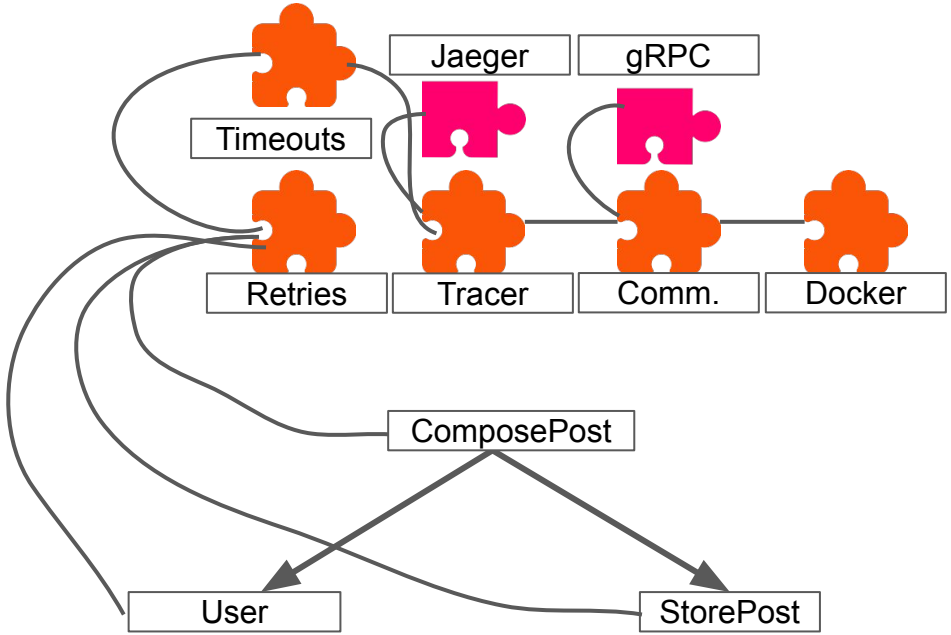
Requires 2 lines of code changes!



Solving Type 1 Metastability Failure

Type 1 Metastability Failure

- **Cause**
  - Increased Load -> More Request Timeouts
  - More Timeouts -> More Retries
  - More Retries -> Load Persists
  - Performance stays degraded!
- **Solution**
  - Add circuit breakers

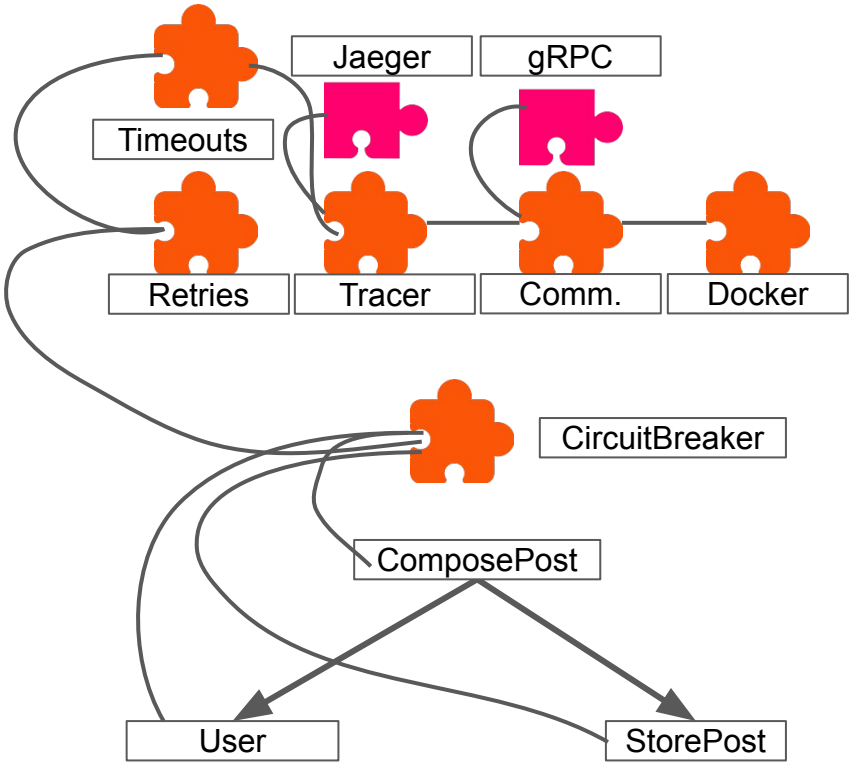


Solving Type 1 Metastability Failure

Type 1 Metastability Failure

- **Cause**
  - Increased Load -> More Request Timeouts
  - More Timeouts -> More Retries
  - More Retries -> Load Persists
  - Performance stays degraded!
- **Solution**
  - Add circuit breakers

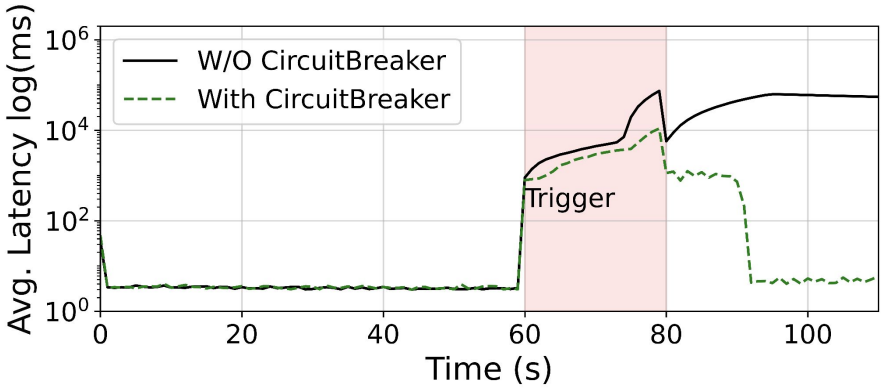
Requires 1 line of code change



Solving Type 1 Metastability Failure

Type 1 Metastability Failure

- **Cause**
  - Increased Load -> More Request Timeouts
  - More Timeouts -> More Retries
  - More Retries -> Load Persists
  - Performance stays degraded!
- **Solution**
  - Add circuit breakers
  - Prevents Load Persistence



Requires 1 line of code change

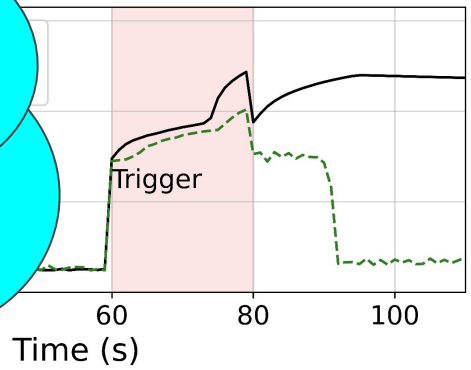
Solving Type 1 Metastability Failure

Type 1 Metastability Failure

- **Cause**
  - Increased Load
    Timeouts
  - More Timeo
  - More Retries
  - Performance
- **Solution**
  - Add circuit breakers
  - Prevents Load Persistence

Requires 1 line of code change

Blueprint reduces the effort required by researchers to perform experiments



Conclusion

Blueprint provides a **toolchain** to **reconfigure, rebuild,** and **deploy**

microservice applications

Conclusion

Blueprint provides a **toolchain** to **reconfigure, rebuild,** and **deploy** **microservice** applications

- **5-7x reduction** in implementation effort
- Modifications require **few lines of code changes**

Conclusion

Blueprint provides a **toolchain** to **Configure, Build,** and **Deploy**

microservice applications

- **5-7x reduction** in implementation effort
- Modifications require **few lines of code changes**

More Info at https://blueprint-uservices.github.io/