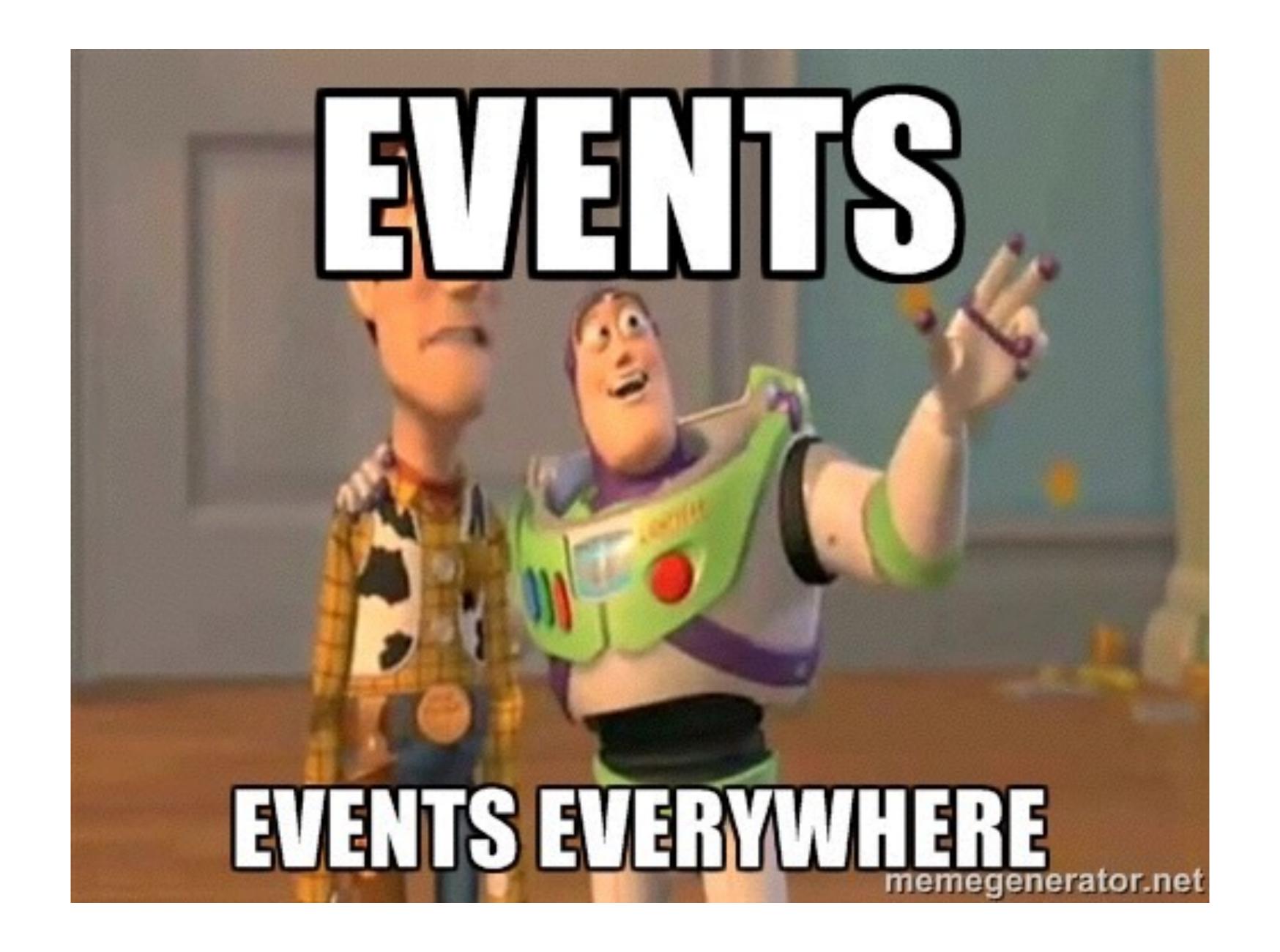


Event sourcing and CQRS from the trenches



SIDNEY SHEK · ARCHITECT · ATLASSIAN · @SIDNEYSHEK

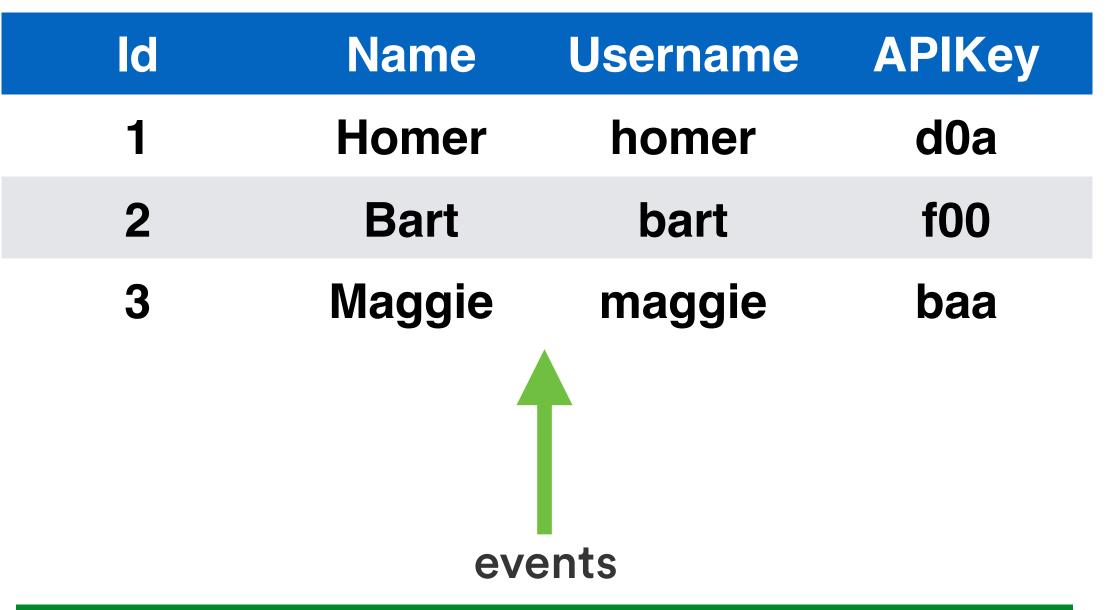




Id	Name	Username	APIKey
1	Homer	homer	d0a
2	Bart	bart	f00
3	Maggie	maggie	baa

ld	Name	Username	APIKey
1	Homer	homer	d0a
2	Bart	bart	f00
3	Lisa Jr	maggie	baa

ld	Name	Username	APIKey
1	Homer	homers	d0a
2	Bart	bart	f00
3	Lisa Jr	maggie	baa



Seq	Event	Time
123	SetUsername(3, Maggie)	0

Id	Name	Username	APIKey
1	Homer	homer	d0a
2	Bart	bart	f00
3	Lisa Jr	maggie	baa
	eve	ents	

Seq	Event	Time
123	SetUsername(3, Maggie)	0
124	SetName(3, Lisa Jr)	10

Id	Name	Username	APIKey
1	Homer	homers	d0a
2	Bart	bart	f00
3	Lisa Jr	maggie	baa
	eve	ents	

Seq	Event	Time
123	SetUsername(3, Maggie)	0
124	SetName(3, Lisa Jr)	10
125	SetUsername(1, homers)	15

users_new

ld	Name	Derived
1	Homer	Homer1
2	Bart	Bart2
3	Lisa Jr	Lisa Jr3

ld	Name	Username	APIKey
1	Homer	homers	d0a
2	Bart	bart	f00
3	Lisa Jr	maggie	baa
	ev	ents	

Seq	Event	Time
123	SetUsername(3, Maggie)	0
124	SetName(3, Lisa Jr)	10
125	SetUsername(1, homers)	15

users_new

ld	Name	Derived
1	Homer	Homer1
2	Bart	Bart2
3	Lisa Jr	Lisa Jr3

users

ld	Name	Username	APIKey		
1	Homer	homers	d0a		
2	Bart	bart	f00		
3	Lisa Jr	maggie	baa		



Seq	Event	Time
123	SetUsername(3, Maggie)	0
124	SetName(3, Lisa Jr)	10
125	SetUsername(1, homers)	15

events



Users, groups and memberships



- Users, groups and memberships
 - Searching for users



- Users, groups and memberships
 - Searching for users
 - Retrieve by email



- Users, groups and memberships
 - Searching for users
 - Retrieve by email



- Users, groups and memberships
 - Searching for users
 - Retrieve by email
 - Incremental synchronisation



- Users, groups and memberships
 - Searching for users
 - Retrieve by email
 - Incremental synchronisation

Audit trails for changes



- Users, groups and memberships
 - Searching for users
 - Retrieve by email
 - Incremental synchronisation

Audit trails for changes

- Highly available
 - Disaster recovery
 - Zero-downtime upgrades



- Users, groups and memberships
 - Searching for users
 - Retrieve by email
 - Incremental synchronisation

Audit trails for changes

High volume low latency reads

- Highly available
 - Disaster recovery
 - Zero-downtime upgrades

Testing with production-like data

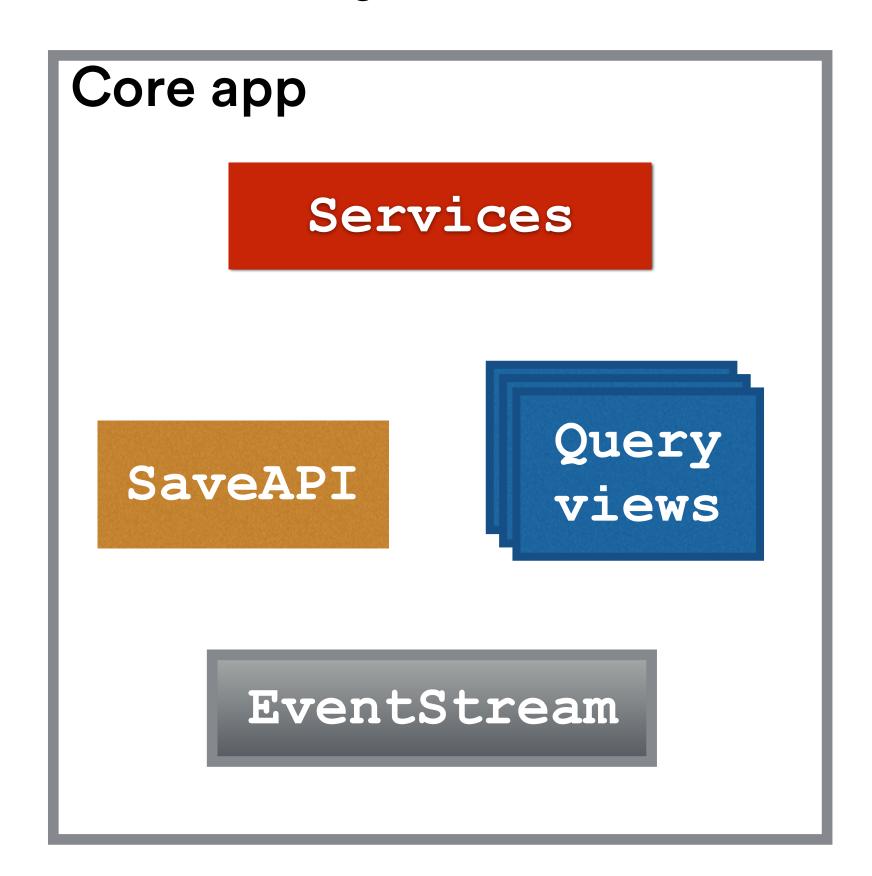




Evolving the architecture

REST calls

e.g. Add User









































REST calls e.g. Add User Core app Services Query Query Commands sync views EventStream **Events Kinesis** Lambda DynamoDB



REST calls e.g. Add User Core app Services Query Query Commands sync views EventStream **ElasticSearch Events Kinesis** Lambda DynamoDB



REST calls e.g. Add User Core app Services Query Query Commands sync views EventStream ElasticSearch **Events** Kinesis Lambda DynamoDB



REST calls e.g. Add User Groups Services Query Query Commands sync views EventStream **Platform Events Kinesis** Lambda DynamoDB



REST calls e.g. Add User Groups Services Query Query Commands sync views EventStream **Platform Events** Users **Kinesis** Lambda DynamoDB



REST calls e.g. Add User Groups Services Query Query Commands sync views EventStream **External Events Platform Events** Users Kinesis **Event** Lambda **Kinesis** DynamoDB Txf

```
{SPR_PUNG,2,4,(NULL),S_PUNCH5,0.0},
                                                                             // S_PUNCH4
                                                                                              // do things to change the game state
                                   (SPR_PUNG, 1, 5, (A_ReFire), S_PUNCH, 0, 0),
                                                                             // S_PUNCH5
                                                                                               while (gameaction != ga_nothing)
                                   {SPR_PISG.0,1,{A_WeaponReady},S_PISTOL,0,0),//S_PISTOL
                                   {SPR_PISG,0,1,{A_Lower},S_PISTOLDOWN,0,0}, // S_PISTOLDOWN {
                                                                                                    switch (gameaction)
                                   {SPR_PISG, 0, 1, {A_Raise}, S_PISTOLUP, 0, 0}, // S_PISTOLUP
                                   (SPR_PISG,0,4,{NULL),S_PISTOL2,0,0), // S_PISTOL1
                                                                                                     case ga_loadlevel:
                                   {SPR_PISG,1,6,{A_FirePistol},S_PISTOL3,0,0};// S_PISTOL2
                                                                                                      G_DoLoadLevel ();
                                   (SPR PISG.2,4,(NULL),S_PISTOL4,0,0), // S_PISTOL3
                                   (SPR PISC.1,5,(A ReFire),S PISTOL,0,0), // S PISTOL4
                                   (SPR_PISE,32768,7,(A_Light1),S_LIGHTDONE,0,0), // S_PISTOLFLASHcase ga_newgame:
                                                                                                       G_DoNewGame ();
                                  (SPR_SHTG,0,1,(A_WeaponReady),S_SGUN,0,0), // S_SGUN
                                   (SPR_SHTG.0.1 (A_Lower),S_SGUNDOWN,0,0), // S_SGUNDOWN
                                                                                                      break;
                                                                                                     case ga_loadgame
                                  (SFR_SHTG,0,1,(A_Raise),S_SGUNUP,0,01, // S_SGUNUP
                                                                                                      G_DoLoadGame ()
                                   (SPR_SHTG,O,3,(NULL),S_SGUN2,O,D), //S_SGUN1
                                                                                                      break,
                                   ISPR SHTG,0,7 (A FireShotgun),S SGUN3,0,0), //S SGUN2
                                   (SPR_SHTG,1,5,(NULL),S_SGUN4,0,0), // S_SGUN3
                                                                                                     case ga_savegame:
G_DoSaveGame 0;
ssy with SPECIAL and commented parts. (SPR_SHTG,2,5,INULL),S_SGUN5,0,0), // S_SGUN4
                                                                                                      break;
                                   (SPR_SHTG.3,4,(NULLI,S_SGUN6,0.0), //S_SGUN5
to make the latest edition work.
                                                                                                     case ga playdemo:
                                   ISPR SHTG,2,5 (NULL),S SGUN7,0.0), //S SGUN6
                                                                                                      G DoPlayDemo ()
                                   (SPR_SHTG,1,5,(NULL),S_SGUNB,0,0), //S_SGUN7
                                                                                                      break,
                                  (SPR_SHTG.0,3 (NULL),S_SGUN9,0,0), // S_SGUN8
                                                                                                     case ga_completed:
                                   (SPR_SHTG,0,7,{A_ReFire},S_SGUN,0,D},//S_SGUN9
                                                                                      // S_SGUNFLASH1_G_DoCompleted ():
                                  (SPR) SHTF, 32768, 4, (A_Light, 1), S_SGUNFLASH2, 0, 0),
                                                                                      W.S. SGLINFLASH2 break;
                                   (SPR_SHTF,32769,3,(A_Light21,S_LIGHTDONE,0,0);
                                   {SPR_SHT2.0,1,(A_WeaponReady},S_DSGUN,0,0),//S_DSGUN
                                                                                                     case ga_victory:
                                                                                                     F_StartFinale ();
                                   (SPR_SHT2,0,1,(A_Lower),S_DSGUNDOWN,0,0); // S_DSGUNDOWN
                                                                                                      break.
                                   (SPR_SHT2,0,1,(A_Raise),S_DSGUNUP,0,0), // S_DSGUNUP
                                                                                                     case ga_worlddone:
                                   (SFR_SHT2,0,3,{NULL1,S_DSGUN2,0,0), // S_DSGUN1
 = shareware )
                                                                                                      G_DoWorldDone ();
                                   (SPR_SHT2,0,7,(A_FireShotgun2),S_DSGUN3,0,0),
                                                                                                      break;
                                   (SPR_SHT2,1,7,(NULL),S_DSGUN4,0,0), // S_DSGUN3
  // only start episode 1 on shareware (SPH_SHT2,2,7,(A_CheckReload),S_DSGUN5,0,0), // S_DSGUN4
                                                                                                     case ga_screenshot:
                                                                                                     M_ScreenShot ();
                                   (SPR_SHT2.3,7;(A_OpenShotgun2),S_DSGUN6,0,0),
                                                                                                       gameaction = ga_nothing;
                                  (SPR_SHT2.4,7,(NULL1,S_DSGUN7,0,0), // S_DSGUN6
                                   (SPR_SHT2,5,7,{A_LoadShotgun2},S_DSGUN8,0,0),
                                                                                                      break;
                                                                                                     case ga_nothing:
                                   ISPR SHT2.6;6;(NULL);S DSGUN9.0;01, // S_DSGUN8
                                   (SPR_SHT2,7;6,{A_CloseShotgun2},S_DSGUN10,0,0). // S_DSGUN9
                                                                                                     break;
                                   (SPR_SHT2,0,5,(A_ReFire),S_DSGUN,0,0), // S_DSGUN10
                                  (SPR_SHT2,1,7,(NULL),S_DSNR2,0,0), // S_DSNR1
                                  {SPR_SHT2,0,3,{NULL1,S_DSGUNDOWN,0,0}, // S_DSNR2 :
                                   {SPR_SHT2,32776,5,{A_Light1},S_DSGUNFLASH2,D,O), // S_DSGUNFgetcommands, check consistancy,
                                   (SFR_SHT2,32777,4,{A_Light2},S_LIGHTDONE,0,0), // S_DSGUM(and build new consistency check
                                                                                             buf = (gametic/ticdup)%BACKUPTICS;
                                   (SPR_CHGG, 0, 1, {A_WeaponReady}, S_CHAIN, 0, 0}, // S_CHAIN
                                   (SPR_CHGG;0,1,(A_Lower),S_CHAINDOV/N,0,0), // S_CHAINDOWN
                                   (SPR_CHGG,0,1,{A_Raise},S_CHAINUP,0,0), // S_CHAINUP for (i=0; i<MAXPLAYERS; i++)
                                   (SFR_CHGG,0,4,{A_FireCGunJ,S_CHAIN2,0,0}, // S_CHAIN1
de != commercial) )
                                   {SPR_CHGG,1,4,(A_FireCGun),S_CHAIN3,0,D}, // S_CHAIN2
                                                                                                    if tplayeringametil)
                                   (SPR_CHGG,1,0,(A_ReFire),S_CHAIN,0,0), // S_CHAIN3
                                   (SFR_CHGE32768,5,(A_Light1),S_LIGHTDONE,0,0), // S_CHAINFLASH1 cmd = &players(i).cmd;
                                   (SFR_CHGF,32769,5,(A_Light2),S_LIGHTDONE,0,0), // S_CHAINFLASH2
                                                                                                     memopy (cmd, &netcmdslillbull, sizeof(ticcmd_tl);
nare II respawnparm )
                                   {SFR_MISG,0,1,(A_WeaponReady},S_MISSILE,0,0), // S_MISSILE
                                   (SPR_MISG,0,1,(A_Lower),S_MISSILEDOWN,0,0), // S_MISSILEDOWN
ters = true;
                                                                                                       if (demoplayback)
                                  (SPR_MISG,0,1,(A_Raise),S_MISSILEUP,0,0), // S_MISSILEUP
                                                                                                           G_ReadDemoTiccmd (cmd);
                                  (SPR_MISG,1,8,(A_GunFlash),S_MISSILE2,0,0), // S_MISSILE1
ters = false:
                                                                                                       if (demorecording)
                                   (SPR_MISG,1,12.(A_FireMissile),S_MISSILE3,0,0),// S_MISSILE2
                                                                                                           G WriteDemoTiccmd (cmd):
sk_nightmare && gameskill != sk_nightmare) MISG,1,0,(A_ReFire),S_MISSILE,0,01, // S_MISSILE3
                                   (SPR_MISF, 32768, 3, (A_Light1), S_MISSILEFLASH2, (), (1), // S_MISSILEFLASH1
                                  (SPR_MISF,32769,4,{NULL},S_MISSILEFLASH3,0,0), // S_MISSILEFLASH# check for turbo cheats
RUN1 : I<=S_SARG_PAIN2 ; i++)
                                   (SPR_MISF,32770,4,fA_Light2),S_MISSILEFLASH4,0,0); // S_MISSILEFLASHif (cmd->forwardmove > TURBOTHRESHOLD
s >>= 1:
                                                                                                          && !(gametic&31) && ((gametic>>5)&3) == i )
BRUISERSHOTI.speed = 20*FRACUNIT; {SPR_MISE,32771,4,{A_Light2},S_LIGHTDONE,0,0}, // S_MISSILEFLASH4
HEADSHOTI.speed = 20*FRACUNIT; (SPR_SAWG,2,4,4A_WeaponReady),S_SAWB,0,01;//S_SAW
TROOPSHOT).speed = 20°FRACUNIT; {SPA_SAWG,3,4,{A_WeaponReady},S_SAW,0,0), // S_SAWB
                                                                                                            static char turbomessage(80);
                                                                                                            extern char *player_namesl41;
                                   {SPR_SAWG,2,1,{A_Lower},S_SAWDOWN,0,0}, // S_SAWDOWN
ghtmare && gameskill == sk_nightmare) {SPR_SAWG,2,1,{A_Raise},S_SAWUP,0,0}, // S_SAWUP
                                                                                                            sprintf (turbomessage, "%s is turbo!" player_namestill);
                                                                                                            players/consoleplayer/.message = turbomessage;
                                   {SPR_SAWG,0,4,{A_Saw},S_SAW2,0,0}; // S_SAW1
                                  {SPR_SAWG,1,4,{A_Sew},S_SAW3,0,0}; // S_SAW2
S_RUN1 : i<=S_SARG_PAIN2 ; i++)
                                   {SPR_SAWG,1,0,{A_ReFire1,S_SAW,0,01, // S_SAW3
s <<= 1.
                                                                                                      if (netgame && !netdemo && !(gametic%tiodup) }
SRUISERSHOTT: speed = 15°FRACUNIT; {SPR_PLSG,0,1,(A_WeaponReadyT,S_PLASMA,0,0}, // S_PLASMA
HEADSHOTI.speed = 10°FRACUNIT:
                                 (SPR_PLSG,0,1,(A_Lower),S_PLASMADOWN,0,0), // S_PLASMADOWN
                                                                                                            if (gametic > BACKUPTICS
TROOPSHOTI.speed = 10*FRACUNIT; {SPR_PLSG,0,1,[A_Raise},S_PLASMAUP,0,0}, // S_PLASMAUP
                                                                                                              && consistancylillbufl != cmd->consistancy)
                                   (SPR_PLSG,0,3,(A_FirePlasma),S_PLASMA2,0,0); // S_PLASMA1
                                  (SPR_PLSG,1,20,(A_ReFire),S_PLASMA,0,0), // S_PLASMA2
                                                                                                              LError ("consistency failure (%) should be %)1, mob_t* target,
                                   (SPR_PLSE,3276B,4,IA_Light11,S_LIGHTDONE,0,0), // S_PLASMAFLASH1
                                   (SPR_PLSF,32769,4,IA_Light1),S_LIGHTDONE,0,0), // S_PLASMAFLASH2
                                                                                                                      cmd->consistancy, consistancyfillbufDpobi_t* emmiter 1
initialized upon first level load
                                   {SPA_BFGG,0,1,[/A_WeaponReady],S_BFG,0,0], // S_BFG
(YERS: i++)
erstate = PST_REBORN;
                                                                                                             consistancylillbufl = playerslil.mo->x;
                                  (SPA_BFGG,0,1,IA_Raise),S_BFGUP,0,0), // S_BFGUP
                                                                                                            eise
      // will be set false if a demo (SFR_BFGG,0,20,(A_BFGsound),S_BFG2,0.0), // S_BFG1
                                  (SPR_BFGC,1,10,(A_GunFlash),S_BFG3,0,0), // S_BFG2
                                                                                                              consistancylillbufl = rndindex;
                                  (SPR_BFGG,1,10,{A_FineBFG},S_BFG4,0,0), // S_BFG3
                                   (SPR_BFGG,1,20,(A_ReFire),S_BFG,0,0), // S_BFG4
                                   (SPR_BFGF,32768,11,(A_Light1),S_BFGFLASH2,0,0), // S_BFGFLASH1
                                   {SPR_BFGF,32769,6,{A_Light2},S_LIGHTDONE,0,0}; // S_BFGFLASH
                                                                                               // check for special buttons
                                   {SPA_BLUD,2,8,(NULL),S_BL0002,0,0), //S_BL0001
                                                                                               for (i=0; i<MAXPLAYERS; i++)
                                   {SPR_BLUD,1,8,{NULL},S_BL0003,0,0}, // S_BL0002
                                   {SPA_BLUD,0,8,(NULL),S_NULL,0,0), // S_BL0003
                                                                                                    if (playeringame(i))
                                   {SPR_PUFF,32768,4,(NULL),S_PUFF2,0,0}, // S_PUFF1
                                   (SPA_PUFF,1,4,(NULL),S_PUFF3,0,0), // S_PUFF2
                                                                                                       if (playerstill.cmd.buttons & BT_SPECIAL)
                                   {SPR_PUFF,2,4,(NULL),S_PUFF4,0,0), // S_PUFF3
or the episode
                                   (SPR_PUFF,3,4,(NULL),S_NULL,0,0), // S_PUFF4
ommercial)
```

G Duneburn III;

Delving into code



Web

Services

Command handlers

Query handlers



Web

Services

Command handlers

command.Op ~> Result

Query handlers

query.Op ~>
 Result

```
case class Save[A](...)
  extends command.Op[A]

case class UserByName[A](...)
  extends query.Op[A]
```



Web

Services

```
def addUser: AppOp[A]
type AppOp[A] = Free[App, A]
type App[A] =
   Coproduct[command.Op, query.Op, A]
```

Command handlers

command.Op ~> Result

Query handlers

```
query.Op ~>
Result
```

```
case class Save[A](...)
  extends command.Op[A]
case class UserByName[A](...)
  extends query.Op[A]
```



```
Web
```

```
post { entity(as[NewUser]) { u =>
   runApi(UserService.addUser(u))
}
val runApi: AppOp[A] => Route
```

```
Services
```

```
def addUser: AppOp[A]
type AppOp[A] = Free[App, A]
type App[A] =
   Coproduct[command.Op, query.Op, A]
```

Command handlers

Query handlers case class Save[A](...)
 extends command.Op[A]
case class UserByName[A](...)
 extends query.Op[A]

```
command.Op ~> Result
```

query.Op ~>
 Result



Commands save events to streams

```
object UserCommandHandler {
 def apply(streams: Streams): command.Op ~> Result =
   new (command.Op ~> Result) {
       def apply[X](a: command.Op[X]): Result[X] =
         a match {
            case Save(u) =>
            case ...
```



Commands save events to streams

```
object UserCommandHandler {
 def apply(streams: Streams): command.Op ~> Result =
    new (command.Op ~> Result) {
       def apply[X](a: command.Op[X]): Result[X] =
         a match {
            case Save(u) =>
              streams.allUsers.save(SingleStreamId, SetName(u.id, u.name))
                .then(streams.userProfile.save(u.id, SetProfile(u))
                .thenReturn(u.id)
            case ...
```



```
class QuerySynchroniser(db: DB) {
  def synchronise(s: Process[Task, StreamEvent]) =
```



```
class QuerySynchroniser(db: DB) {
  def synchronise(s: Process[Task, StreamEvent]) =
    (s through writeEvent).run
  private val writeEvent: Channel[Task, StreamEvent, Unit] =
    Process.constant { e: StreamEvent =>
```



```
class QuerySynchroniser(db: DB) {
  def synchronise(s: Process[Task, StreamEvent]) =
    (s through writeEvent).run
  private val writeEvent: Channel[Task, StreamEvent, Unit] =
    Process.constant { e: StreamEvent =>
      for {
        r1 <- StreamState.transitionSeq(db, e.id.key, e.id.seq)
      } yield ()
```



```
class QuerySynchroniser(db: DB) {
 def synchronise(s: Process[Task, StreamEvent]) =
    (s through writeEvent).run
 private val writeEvent: Channel[Task, StreamEvent, Unit] =
    Process.constant { e: StreamEvent =>
      for {
        r1 <- StreamState.transitionSeq(db, e.id.key, e.id.seq)
          <- if (r1.success) db.process(e) else db.rollback
     } yield ()
```



...that you pass it an event stream

```
class KinesisProcessor(qs: QuerySynchroniser) extends IRecordProcessor {
  override def processRecords(rs: ProcessRecordInput) = {
  }
}
```



...that you pass it an event stream

```
class KinesisProcessor(qs: QuerySynchroniser) extends IRecordProcessor {
  override def processRecords(rs: ProcessRecordInput) = {
     // Either emit events from Kinesis, or read events from Dynamo
     val stream = Process.emitAll { rs.getRecords. ... }
}
```



...that you pass it an event stream

```
class KinesisProcessor(qs: QuerySynchroniser) extends IRecordProcessor {
  override def processRecords(rs: ProcessRecordInput) = {
     // Either emit events from Kinesis, or read events from Dynamo
     val stream = Process.emitAll { rs.getRecords. ... }

     qs.synchronise(stream).attemptRunFor(...)
}
```





Events as an API

Insert / Update Delta vs 'Set' events



Insert / Update Delta vs 'Set' events

AddUser(id, name, email1)



Insert / Update Delta vs 'Set' events

AddUser(id, name, email1)

UpdateUser(id, email = Some(email))



AddUser(id, name, email1)

UpdateUser(id, email = *Some*(email))

'Set' events

SetUserName(id, name)
SetUserEmail(id, email1)



AddUser(id, name, email1)

UpdateUser(id, email = Some(email))

'Set' events

SetUserName(id, name)

SetUserEmail(id, email1)

SetUserEmail(id, email2)



AddUser(id, name, email1)

UpdateUser(id, email = Some(email))

Fits nicely with CRUD + PATCH

'Set' events

SetUserName(id, name)
SetUserEmail(id, email1)

SetUserEmail(id, email2)



AddUser(id, name, email1)

UpdateUser(id, email = *Some*(email))

Fits nicely with CRUD + PATCH

Assume insert before update

'Set' events

SetUserName(id, name)

SetUserEmail(id, email1)

SetUserEmail(id, email2)



AddUser(id, name, email1)

UpdateUser(id, email = *Some*(email))

Fits nicely with CRUD + PATCH

Assume insert before update

'Set' events

SetUserName(id, name)

SetUserEmail(id, email1)

SetUserEmail(id, email2)

Encourages idempotent processing



AddUser(id, name, email1)

UpdateUser(id, email = *Some*(email))

Fits nicely with CRUD + PATCH

Assume insert before update

'Set' events

SetUserName(id, name)

SetUserEmail(id, email1)

SetUserEmail(id, email2)

Encourages idempotent processing

Single code path for query sync



AddUser(id, name, email1)

UpdateUser(id, email = Some(email))

Fits nicely with CRUD + PATCH

Assume insert before update

'Set' events

SetUserName(id, name)

SetUserEmail(id, email1)

SetUserEmail(id, email2)

Encourages idempotent processing

Single code path for query sync

Minimally sized events to avoid conflict



AddUser(id, name, email1)

UpdateUser(id, email = *Some*(email))

Fits nicely with CRUD + PATCH

Assume insert before update

'Set' events

SetUserName(id, name)
SetUserEmail(id, email1)

SetUserEmail(id, email2)

Encourages idempotent processing

Single code path for query sync

Minimally sized events to avoid conflict



Single stream

Multiple streams



Single stream

Transactions and consistent data resolution

Multiple streams



Single stream

Transactions and consistent data resolution

Multiple streams

Sharding for throughput



Rules for splitting streams

1. Place independent events on different streams



Rules for splitting streams

- 1. Place independent events on different streams
- 2. Split streams by event type and unique Id



Rules for splitting streams

- 1. Place independent events on different streams
- 2. Split streams by event type and unique Id
- 3. Identify the 'transactions' you really need



Rules for splitting streams

- 1. Place independent events on different streams
- 2. Split streams by event type and unique Id
- 3. Identify the 'transactions' you really need
- 4. Use hierarchical streams to maximise number of streams



Rules for splitting streams

- 1. Place independent events on different streams
- 2. Split streams by event type and unique Id
- 3. Identify the 'transactions' you really need
- 4. Use hierarchical streams to maximise number of streams
- 5. Splitting and joining streams later is possible





Let go of transactions and consistency



No guaranteed order between events on different streams



- No guaranteed order between events on different streams
- A field should only be updated by a single event stream



- No guaranteed order between events on different streams
- A field should only be updated by a single event stream
- No foreign key constraints



- No guaranteed order between events on different streams
- A field should only be updated by a single event stream
- No foreign key constraints
- Unique or data constraints 'enforced' on write



User: homer (id 4)

All Users: Seq 100

User 4: Seq 23





User: homer (id 4)

All Users: Seq 100

User 4: Seq 23





User: homer (id 4)

All Users: Seq 100

User 4: Seq 23



Returned on read and write via ETag

Pass as request header for:



User: homer (id 4)

All Users: Seq 100

User 4: Seq 23



- Pass as request header for:
 - Condition write ('transaction')



User: homer (id 4)

All Users: Seq 100

User 4: Seq 23



- Pass as request header for:
 - Condition write ('transaction')
 - Force query view update ('consistency')



User: homer (id 4)

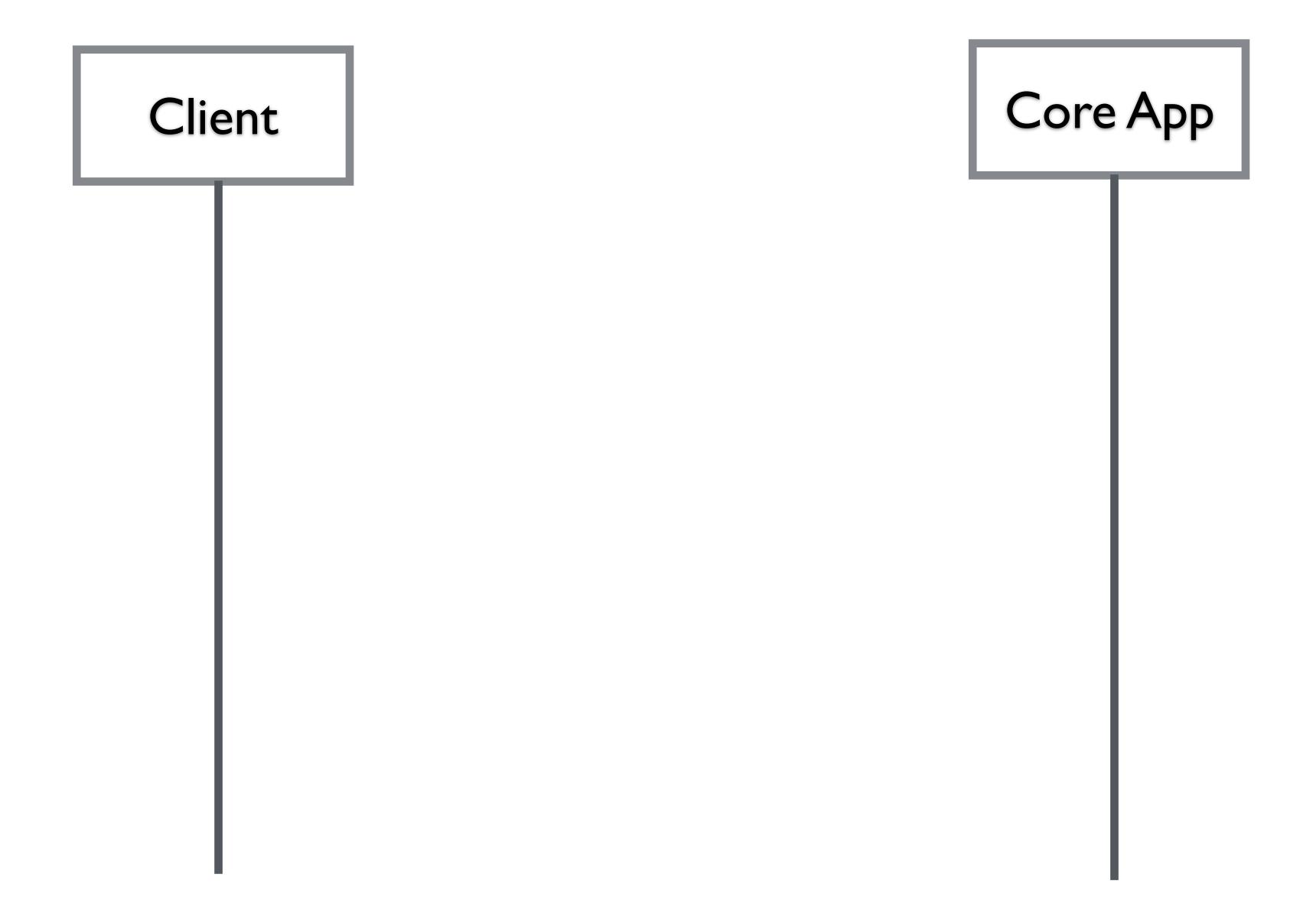
All Users: Seq 100

User 4: Seq 23

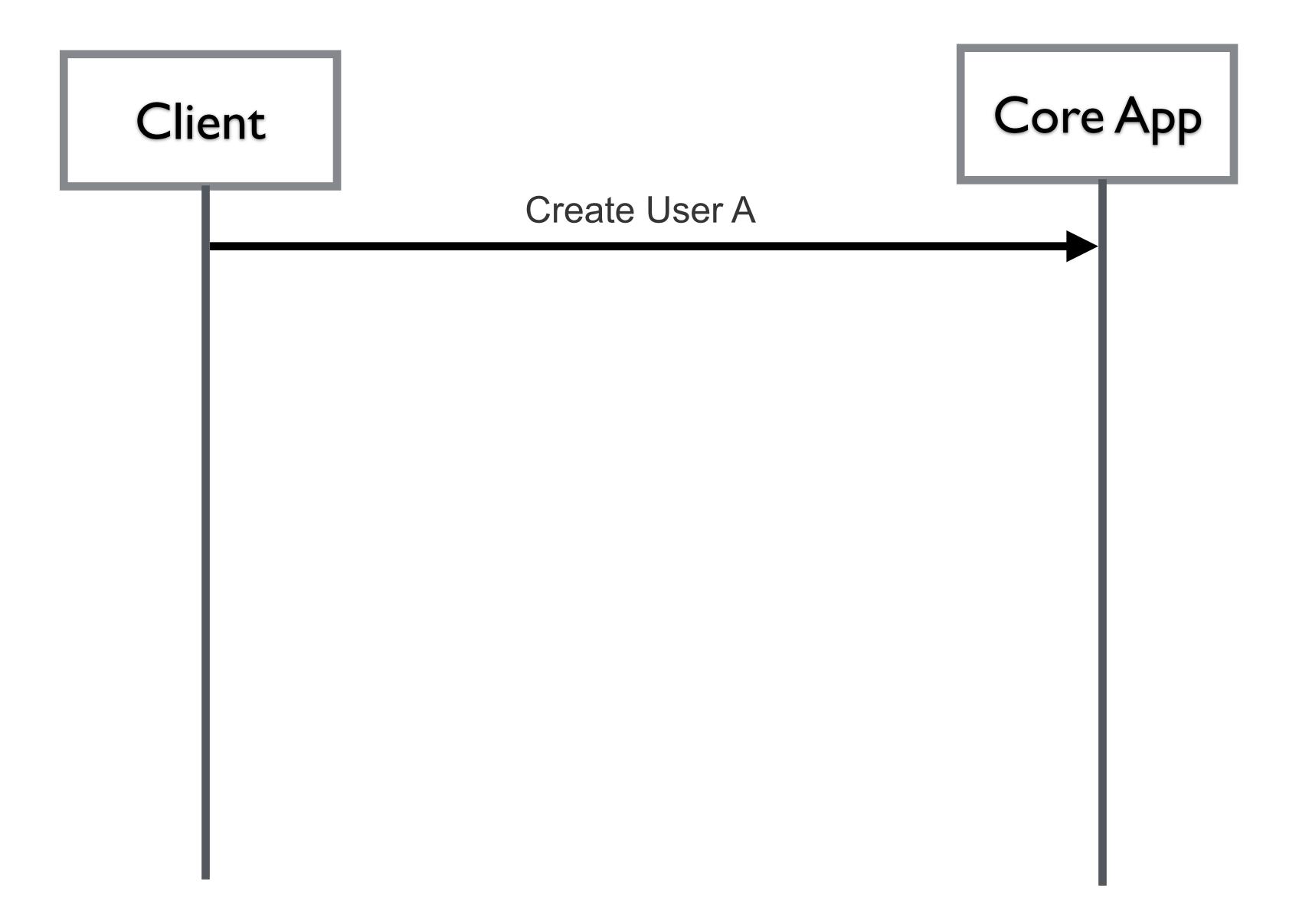


- Pass as request header for:
 - Condition write ('transaction')
 - Force query view update ('consistency')
 - Caching

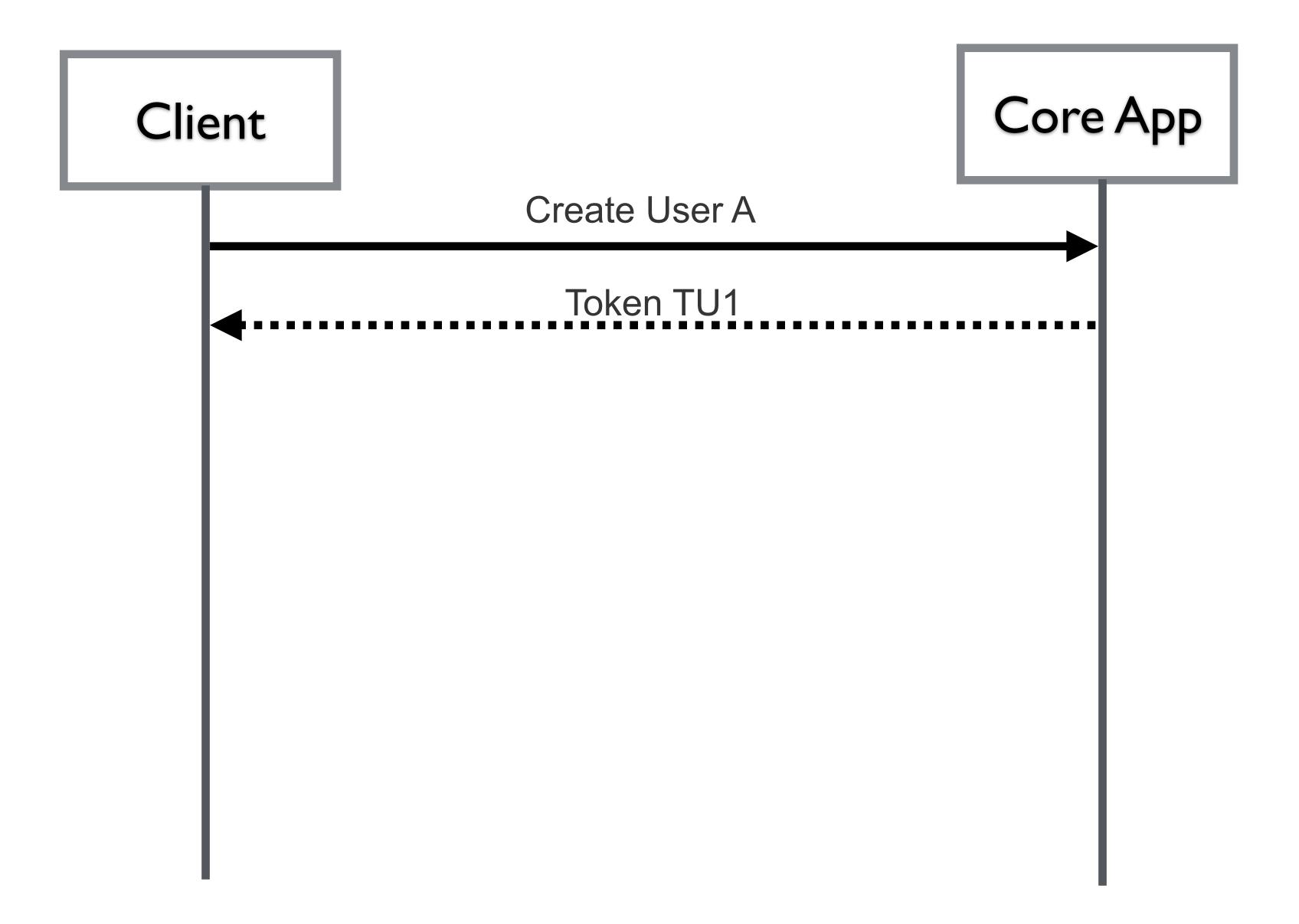




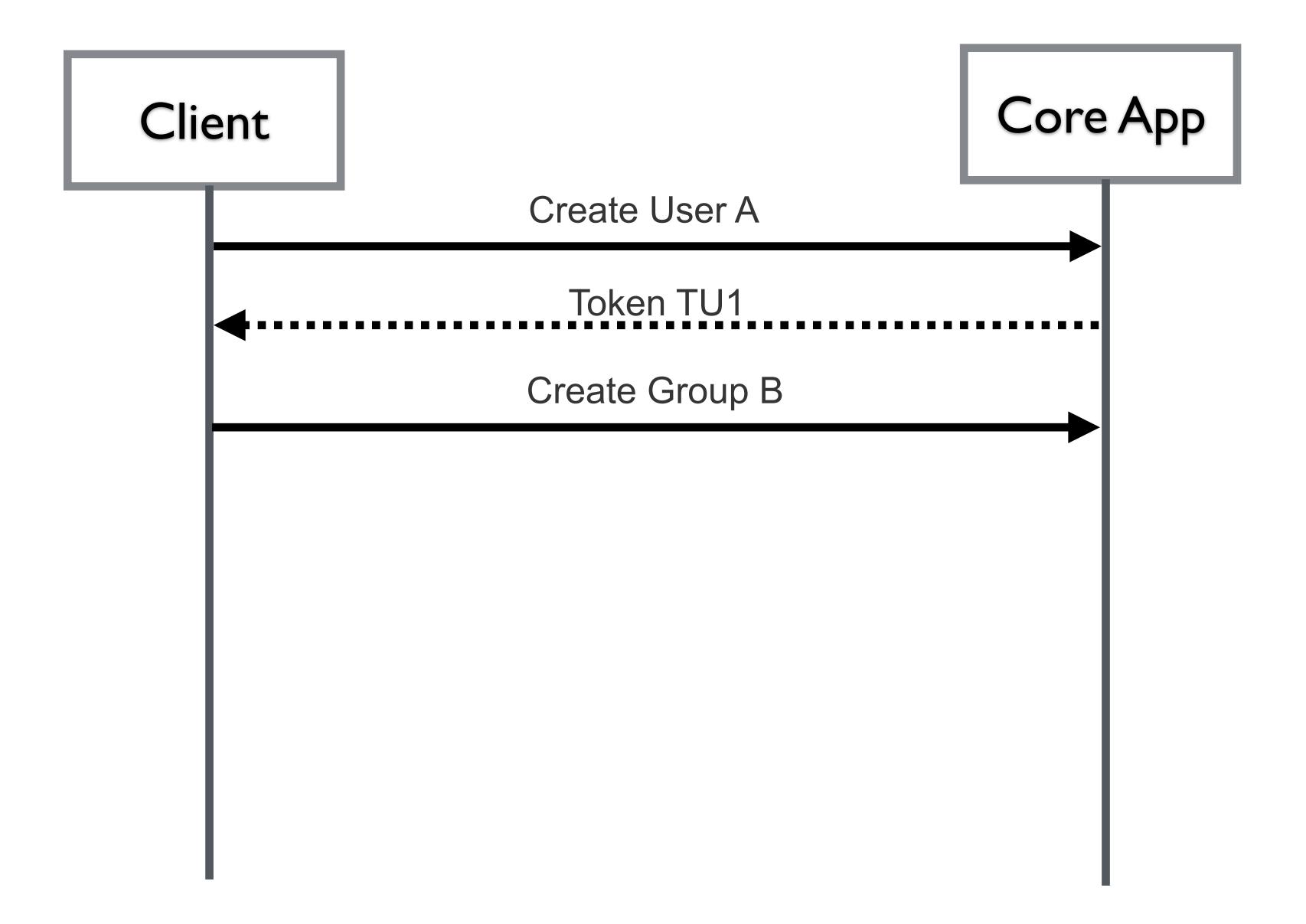




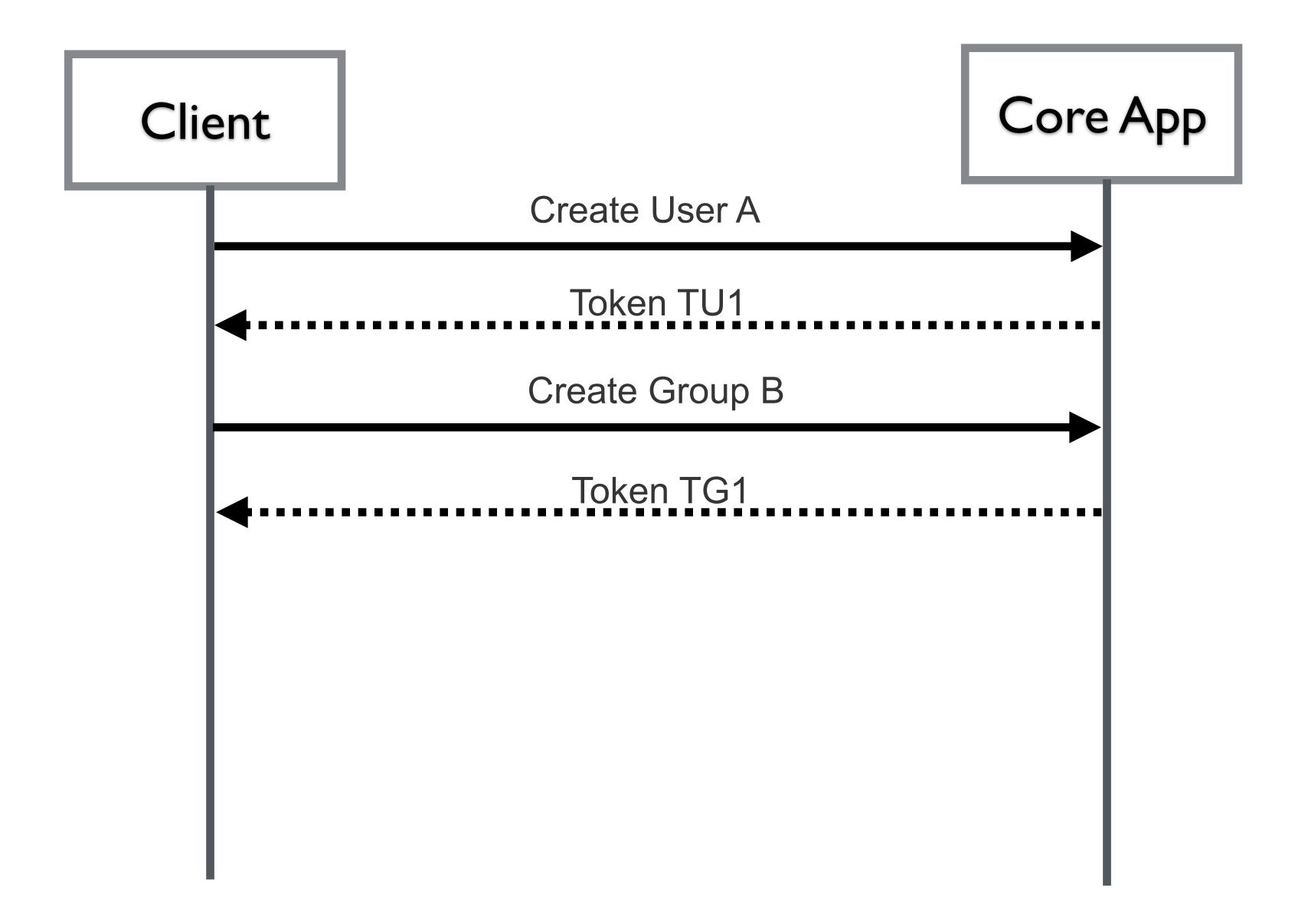




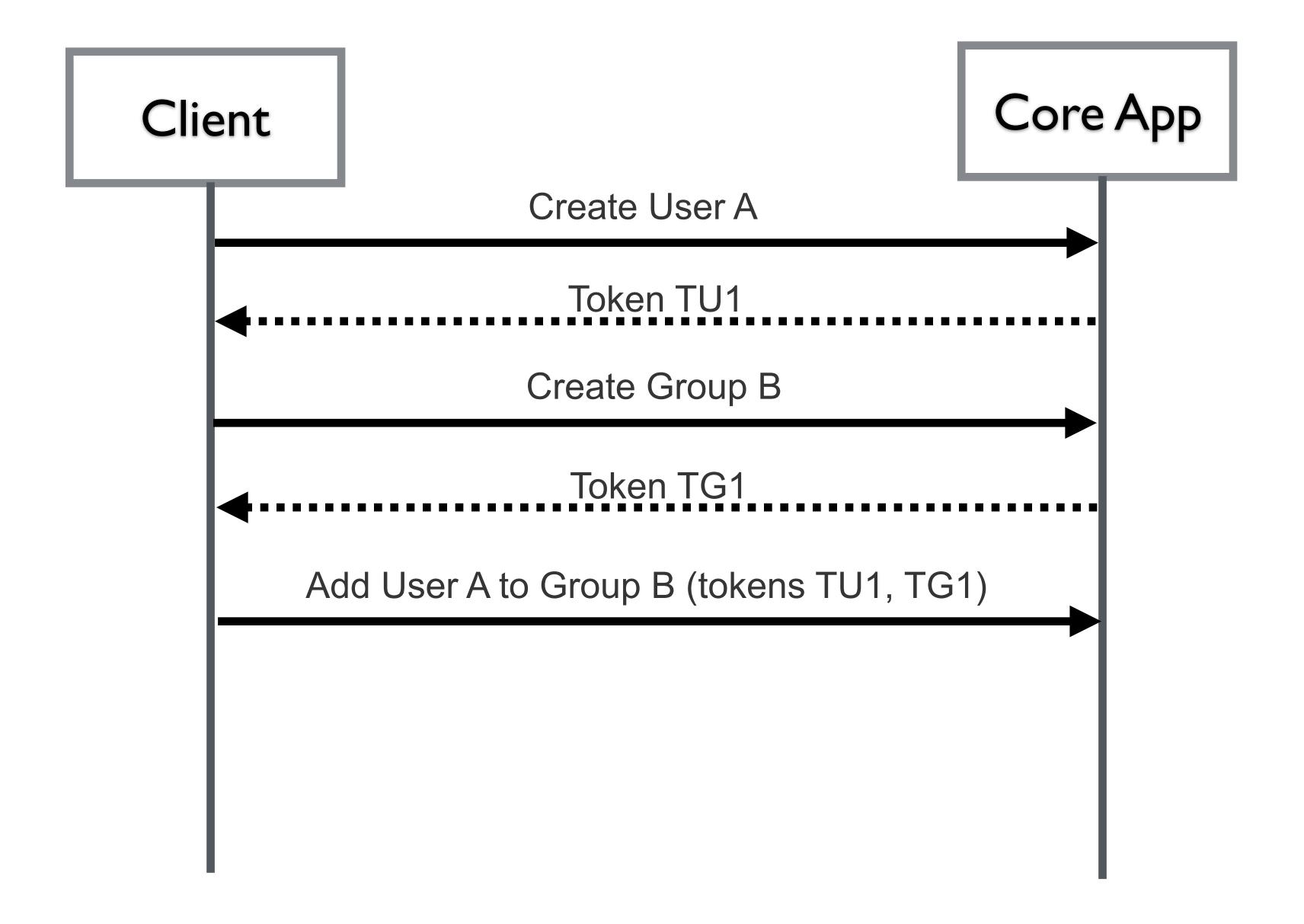




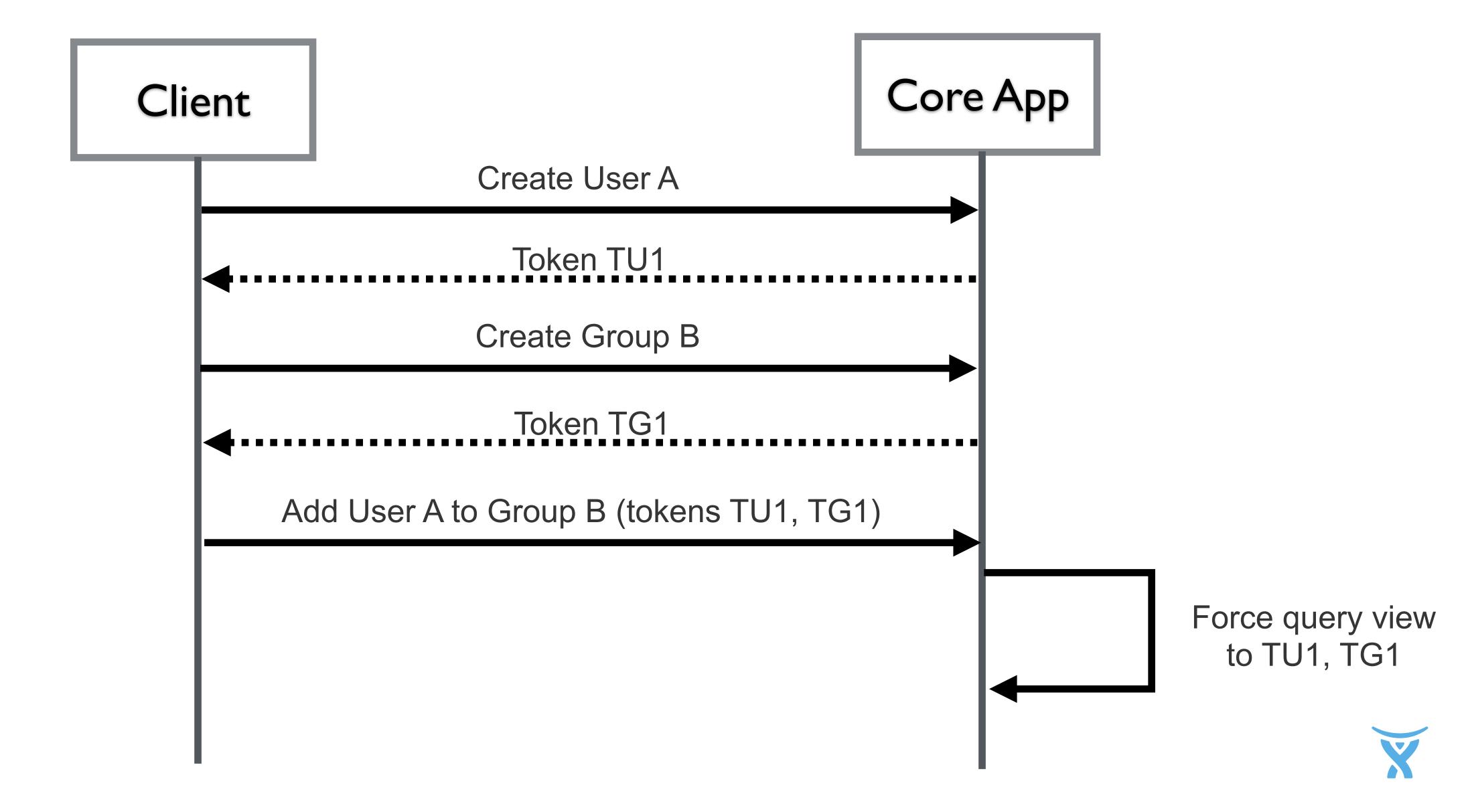


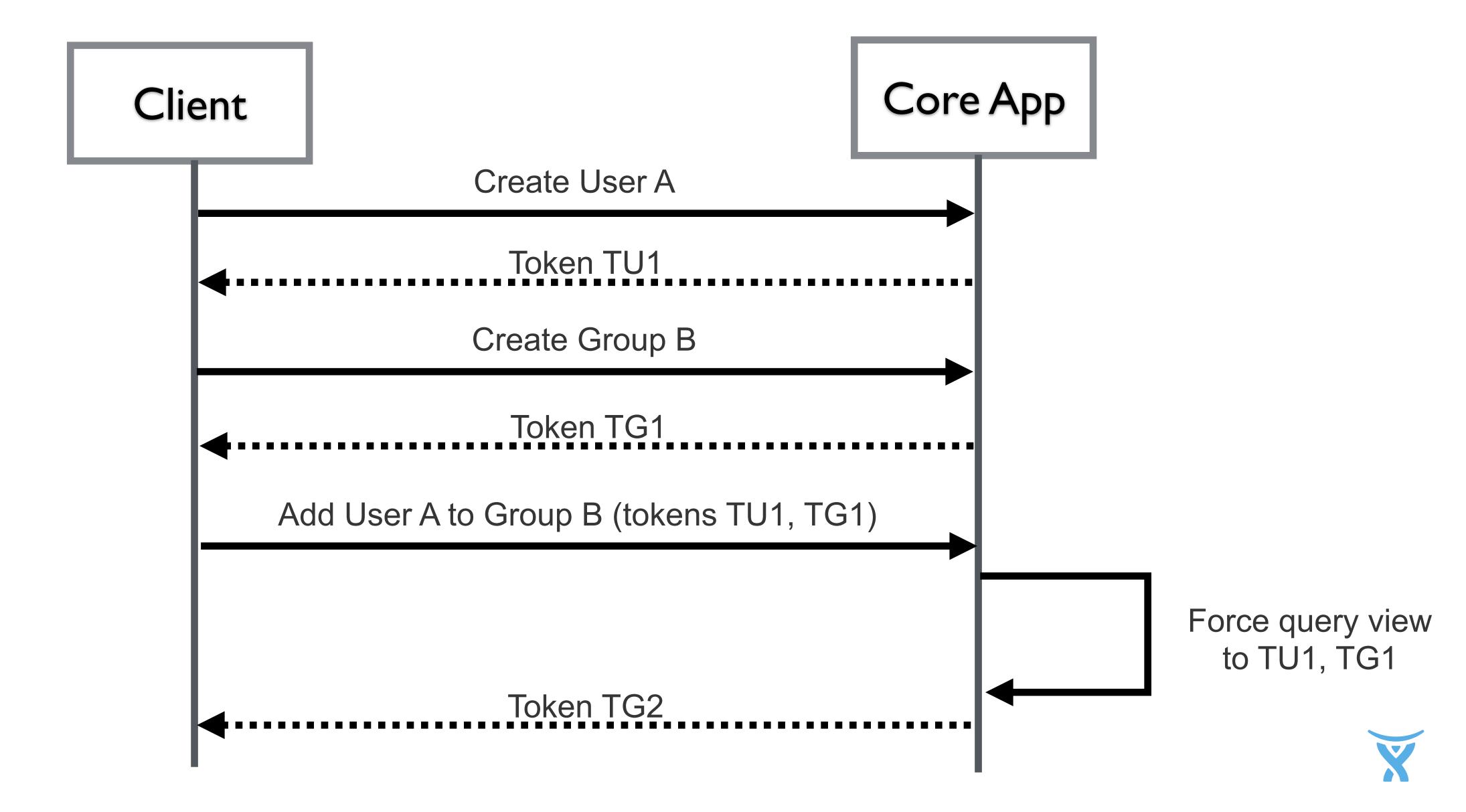














Start small and challenge everything!



Start small and challenge everything!

Incremental architecture for incremental demos



Start small and challenge everything!

Incremental architecture for incremental demos

Think "Events as an API"



Start small and challenge everything!

Incremental architecture for incremental demos

Think "Events as an API"

Accept weaker transactions and eventual consistency



"We should using event sourcing more than we do"

Martin Fowler (very loosely paraphrased)



event sourcing lib: bitbucket.org/atlassianlabs/eventsrc

