Go Agent v0.1.12

배포일: 2022-05-26

베타 - 업데이트

alpine linux 를 지원합니다.

설치 안내

whatap-agent.tar.gz을 다운받고 '/' 디렉터리 기준으로 압축을 해제합니다. /usr/whatap/agent 디렉터리에 모니터링 설치 파일이 생성됩니다.

```
wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/x86_64/whatap-agent.tar.gz
tar -xvzf whatap-agent.tar.gz -C /
```

whatap-agent 실행

```
/usr/whatap/agent/whatap-agent
    Default restart
    Command start, stop, restart, version

## 버전 확인
# /usr/whatap/agent/whatap-agent version
0.8.5.20201209

## 실행 확인
# ps -elf | grep whatap
  103 root      0:05 ./whatap_agent_static -t=4
```

github.com/go-chi/chi 라이브러리 지원

chi 프레임워크의 웹 트랜잭션을 추적합니다. `Use` 함수를 통해 미들웨어를 등록하여 추적합니다.

설치 안내

```
import (
  "github.com/go-chi/chi"
  "github.com/whatap/go-api/trace"
  "github.com/whatap/go-api/instrumentation/github.com/go-chi/chi/whatapchi"
)

func main() {
  config := make(map[string]string)
  trace.Init(config)
  defer trace.Shutdown()

  r := chi.NewRouter()

  // whatapchi의 middleware를 등록합니다.
  r.Use(whatapchi.Middleware)

  r.Get("/", func(w http.ResponseWriter, r *http.Request) {
     fmt.Println("Response -", r.Response)
})
```

예제 참조

github.com/go-gorm/gorm 라이브러리 지원

gorm v2 프레임워크를 통해 처리되는 DB Connection 및 SQL을 추적합니다.

whatapgorm 사용 방식

`gorm.Open` 함수 대신에 `whatapgorm.OpenWithContext` 함수를 사용합니다. 전달하는 context는 내부에 whatap TraceCtx를 포함해야 하며 trace 패키지의 `Start` 함수를 통해 생성 할 수 있습니다.

Install guide

```
import (
  "net/http"

  "github.com/whatap/go-api/instrumentation/github.com/go-gorm/gorm/whatapgorm"
  "github.com/whatap/go-api/trace"
  "gorm.io/driver/sqlite"
  "gorm.io/gorm"
)

func main() {
  whatapConfig := make(map[string]string)
```

```
  trace.Init(whatapConfig)
  defer trace.Shutdown()

  http.HandleFunc("/InsertAndDelete", func(w http.ResponseWriter, r *http.Request) {
    // Context를 생성합니다.
    ctx, _ := trace.StartWithRequest(r)
    defer trace.End(ctx, nil)

    // whatapgorm을 통해 DB를 연결합니다.
    db, err := whatapgorm.OpenWithContext(sqlite.Open("test.db"), &gorm.Config{}, ctx)
    if err != nil {
      panic("Db 연결에 실패하였습니다.")
    }

    for i := 0; i < 100; i++ {
      db.Create(&Product{Code: i, Price: i * 100})
    }

    db.Unscoped().Delete(&Product{}, "Code >= ? AND Code < ?", 0, 100)
  })

  _ = http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
}
```

whatapsql 사용 방식

gorm은 공식 지원하는 sqlite, mysql, postgres, sqlserver 외에도 dialect interface 기반으로 작성된 dirver에 대해서 호환 가능합니다.

관련 링크 : gorm driver

Install guide

```
import (
  "net/http"
  "github.com/whatap/go-api/instrumentation/database/sql/whatapsql"
  "github.com/whatap/go-api/trace"
  "gorm.io/driver/mysql"
  "gorm.io/gorm"
)

func main() {
  whatapConfig := make(map[string]string)
  trace.Init(whatapConfig)
  defer trace.Shutdown()

  http.HandleFunc("/WhatapDriverTest", func(w http.ResponseWriter, r *http.Request) {
    // Context를 생성합니다.
    ctx, _ := trace.StartWithRequest(r)
```

```
    defer trace.End(ctx, nil)

    // whatapsql driver로 db connection을 생성합니다.
    dbConn, err := whatapsql.OpenContext(ctx, "mysql", dataSource)

    // 기 생성된 connection을 통해 gorm에 연결합니다.
    db, err := gorm.Open(mysql.New(mysql.Config{Conn: dbConn}), &gorm.Config{})
    if err != nil {
      panic("Db 연결에 실패하였습니다.")
    }

    for i := 0; i < 100; i++ {
      db.Create(&Product{Code: i, Price: i * 100})
    }
  })

  _ = http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
}
```

참조 예제

github.com/jinzhu/gorm 라이브러리 지원

gorm v1 프레임워크를 통해 처리되는 DB Connection 및 SQL을 추적합니다.

whatapgorm 사용 방식

`gorm.Open` 함수 대신에 `whatapgorm.OpenWithContext` 함수를 사용합니다. 전달하는 context는 내부에 whatap TraceCtx를 포함해야 하며 trace 패키지의 Start 함수를 통해 생성 할 수 있습니다.

Install guide

```
import (
  "net/http"

  "github.com/whatap/go-api/instrumentation/github.com/go-gorm/gorm/whatapgorm"
  "github.com/whatap/go-api/trace"
  _ "github.com/mattn/go-sqlite3"
  "github.com/jinzhu/gorm"
)

func main() {
  whatapConfig := make(map[string]string)
  trace.Init(whatapConfig)
  defer trace.Shutdown()

  http.HandleFunc("/InsertAndDelete", func(w http.ResponseWriter, r *http.Request) {
```

```
    // Context를 생성합니다.
    ctx, _ := trace.StartWithRequest(r)
    defer trace.End(ctx, nil)

    // whatapgorm을 통해 DB를 연결합니다.
    db, err := whatapgorm.OpenWithContext(ctx, "sqlite3", "test.db")
    defer db.Close()
    if err != nil {
      trace.Error(ctx, err)
      panic("Gorm Open Fail")
    }

    for i := 0; i < 100; i++ {
      db.Create(&Product{Code: i, Price: i * 100})
    }

    db.Unscoped().Delete(Product{}, "Code >= ? AND Code < ?", 0, 100)
  })

  _ = http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
}
```

whatapsql 사용 방식

gorm은 공식 지원하는 sqlite, mysql, postgres, sqlserver 외에도 dialect interface 기반으로 작성된 dirver에 대해서 호환 가능합니다.

관련 링크 : gorm driver

Install guide

```
import (
  "net/http"
  "github.com/whatap/go-api/instrumentation/database/sql/whatapsql"
  "github.com/whatap/go-api/trace"
  "github.com/jinzhu/gorm"
_ "github.com/go-sql-driver/mysql"
)

func main() {
  whatapConfig := make(map[string]string)
  trace.Init(whatapConfig)
  defer trace.Shutdown()

  http.HandleFunc("/WhatapDriverTest", func(w http.ResponseWriter, r *http.Request) {
    // Context를 생성합니다.
    ctx, _ := trace.StartWithRequest(r)
    defer trace.End(ctx, nil)
```

```
    // whatapsql driver로 db connection을 생성합니다.
    var conn gorm.SQLCommon
    var err error
    conn, err = whatapsql.OpenContext(ctx, "mysql", dataSource)
    if err != nil {
      trace.Error(ctx, err)
      panic("Whatapsql Open Fail")
    }

    // 기 생성된 connection을 통해 gorm에 연결합니다.
    db, err := gorm.Open("mysql", conn)
    if err != nil {
      trace.Error(ctx, err)
      panic("Gorm Open Fail")
    }
    for i := 0; i < 100; i++ {
      db.Create(&Product{Code: i, Price: i * 100})
    }
  })

  _ = http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
}
```

참조 예제

github.com/gomodule/redigo 라이브러리 지원

redigo 프레임워크를 통해 redis에 전달되는 명령을 추적합니다. `redis.Dial` 대신에 `whatapredigo.DialContext`를 함수를 사용합니다.

Install guide

```
import (
  "context"
  "net/http"

  "github.com/gomodule/redigo/redis"
  "github.com/whatap/go-api/instrumentation/github.com/gomodule/redigo/whatapredigo"
  "github.com/whatap/go-api/trace"
)

func main() {
  whatapConfig := make(map[string]string)
  trace.Init(whatapConfig)
  defer trace.Shutdown()

  http.HandleFunc("/SetAndGetWithDialContext", func(w http.ResponseWriter, r *http.Request) {
    // Context를 생성합니다.
```

```
    ctx, _ := trace.StartWithRequest(r)
    defer trace.End(ctx, nil)

    // whtapredigo를 통해 redis connection을 생성합니다.
    conn, err := whatapredigo.DialContext(ctx, "tcp", "127.0.0.1:6379")
    if err != nil {
      trace.Error(ctx, err)
      return
    }
    defer conn.Close()

    _, err = conn.Do("SET", "DataKey", "DataValue")
    if err != nil {
      trace.Error(ctx, err)
      return
    }

    data, err := redis.Bytes(conn.Do("GET", "DataKey"))
    if err != nil {
      trace.Error(ctx, err)
      return
    }
  })
}
```

참조 예제

github.com/shopify/sarama 라이브러리 지원

sarama 프레임워크를 통해서 처리되는 kafka produce, consume 이벤트를 추적합니다.

async producer 추적

whatapsarama의 Interceptor를 통해 async producer 정보를 추적합니다. Producer Message 생성시 Ctx 관련 정보를 Metadata를 통해 전달하면 Multi Transaction으로 연결 됩니다.

Install guide

```
import (
  "context"
  "net/http"
  "github.com/Shopify/sarama"
  "github.com/whatap/go-api/instrumentation/github.com/Shopify/sarama/whatapsarama"
  "github.com/whatap/go-api/trace"
)

func main() {
```

```
  config := sarama.NewConfig()
  brokers := []string{"127.0.0.1:9092"} //config kafka broker IP/Port

  // whatapsarama의 인터럽트를 config에 등록합니다.
  interceptor := whatapsarama.Interceptor{Brokers: brokers}
  config.Producer.Interceptors = []sarama.ProducerInterceptor{&interceptor}

  whatapConfig := make(map[string]string)
  trace.Init(whatapConfig)
  defer trace.Shutdown()

  // Prdoducer 생성시 config를 전달합니다.
  producer, err := sarama.NewAsyncProducer(brokers, config)
  consumerOffset := sarama.OffsetOldest
  if err != nil {
    panic(err)
  }
  defer func() {
    if err := producer.Close(); err != nil {
      panic(err)
    }
  }()

  http.HandleFunc("/AsyncProduceInput", func(w http.ResponseWriter, r *http.Request) {
    ctx, _ := trace.StartWithRequest(r)
    defer func() {
      trace.End(ctx, nil)
    }()
    msg := &sarama.ProducerMessage{
      Topic:    "tmp-topic",
      Key:      sarama.StringEncoder("Data Key"),
      Value:    sarama.StringEncoder("Data Value"),
      Metadata: trace.GetMTrace(ctx),
    }
    producer.Input() <- msg //error check

  })

  _ = http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
}
```

consumer 추적

whatapsarama의 Interceptor를 통해 consumer 정보를 추적합니다. Produce에서 전달된 Message 기준으로 Multi Transaction으로 연결 됩니다.

Install guide

```
import (
```

```
  "context"
  "flag"
  "fmt"
  "net/http"
  "text/template"

  "github.com/Shopify/sarama"
  "github.com/whatap/go-api/instrumentation/github.com/Shopify/sarama/whatapsarama"
  "github.com/whatap/go-api/trace"
)

func main() {
  config := sarama.NewConfig()
  brokers := []string{"127.0.0.1:9092"} //config kafka broker IP/Port

  // whatapsarama의 인터럽트를 config에 등록합니다.
  interceptor := whatapsarama.Interceptor{Brokers: brokers}
  config.Consumer.Interceptors = []sarama.ConsumerInterceptor{&interceptor}

  // consume 1회당1tx
  // Consumer 생성시 config를 전달합니다.
  consumer, err := sarama.NewConsumer(brokers, config)
  topic := "tmp-topic"

  partitions, _ := consumer.Partitions(topic)
  consume, _ := consumer.ConsumePartition(topic, partitions[0], consumerOffset)

  if consume == nil {
    fmt.Println("consume nil")
    return
  }

  go func() {
    for {
      select {
      case msg := <-consume.Messages():
        fmt.Println(msg)
      case consumerError := <-consume.Errors():
        fmt.Println("error", consumerError)
        return
      }
    }
  }()
}
```

[참조 예제]