TaskFlow Pro – הארכיטקטורה של הפרויקט

1. High-Level Layers and Responsibilities

1. **Frontend Layer**
   - **Static Hosting (S3; optional CloudFront)**: Serves the SPA assets (HTML/CSS/JS).
   - **Browser SPA**: Handles UI, authentication flows (via Cognito), and REST calls to backend API.
2. **Authentication Layer (Amazon Cognito)**
   - **User Pool**: Manages sign-up/sign-in, issues JWTs.
   - **User Groups**: users (regular) vs. admins (elevated).
   - **Lambda Triggers**:
     - *Pre-Signup*: Validate email domain.
     - *Post-Confirmation*: Auto-add new users to users group.
3. **API Layer (API Gateway)**
   - **REST API** with Cognito authorizer: validates JWT on each request.
   - **Resources/Methods**:
     - /tasks (GET, POST),
     - /tasks/{taskId} (PUT, DELETE),
     - /tasks/bulk-import (POST),
     - /admin/analytics (GET).
   - **Integration**: Lambda proxy to business-logic Lambdas; CORS enabled for SPA origin.
4. **Business Logic Layer (AWS Lambda functions)**
   - **TaskHandler**: CRUD and enqueue bulk-import messages.
   - **SQSProcessor**: Processes bulk-import messages, writes tasks in batches.
   - **Analytics**: Periodically (EventBridge) or on-demand computes statistics.
   - **Cognito Triggers**: PreSignupValidation and AutoAssignGroup Lambdas.
   - **Config Access**: Read SNS topic ARN & SQS URL from Parameter Store (cached per invocation).
5. **Data Layer (DynamoDB)**
   - **Tasks Table**
     - PK: userId; SK: taskId.
     - Attributes: title, description, priority, dueDate, status, createdAt, userEmail, etc.
     - Access: Query by userId for user operations; Scan or auxiliary index for admin or analytics.
   - **Analytics Table**
     - PK: date (YYYY-MM-DD); SK: metric.
     - Stores daily metrics (counts, distributions, averages).
6. **Messaging Layer**
   - **SNS Topic**: TaskFlow-Notifications.
     - Events: new task, bulk-import start/completion.
     - Subscribed admin emails receive notifications.
   - **SQS Queue**: TaskFlow-ProcessingQueue.
     - Holds messages for bulk-import; triggers SQSProcessor Lambda.

        - Batch size up to 10; visibility timeout configured; optional DLQ for failures.
7. **Configuration & Scheduling**
    - **Parameter Store**: single JSON parameter (e.g., /taskflow/config) with SNS ARN and SQS URL. Lambdas read this at startup.
    - **EventBridge Rule**: daily schedule triggers Analytics Lambda at midnight.
8. **Monitoring & Audit**
    - **CloudWatch Logs/Metrics**: Lambdas log actions; metrics on invocations/errors/durations. Alarms can be set for error spikes or SQS backlog.
    - **CloudTrail**: Captures API calls for auditing resource changes and operations.

2. Component Interactions and Data Flows

2.1 Task Creation (Regular Flow)

1. **Browser**: User submits “create task” form.
2. **API Gateway**: Receives POST /tasks with JWT, validates via Cognito authorizer.
3. **TaskHandler Lambda**:
    - Extracts userId and email from JWT claims.
    - Reads config (SNS ARN) from Parameter Store (cached).
    - Generates taskId, constructs item, writes to DynamoDB Tasks table.
    - Publishes “new task” message to SNS.
    - Returns HTTP 201 with task details.
4. **SNS**: Delivers email notification to subscribed admins.

2.2 Task Retrieval / Update / Delete

- **GET /tasks**: TaskHandler queries DynamoDB by partition key userId; if admin, may scan or use index.
- **PUT /tasks/{taskId}** and **DELETE /tasks/{taskId}**: TaskHandler checks ownership (userId matches) or allows admin; performs UpdateItem or DeleteItem; may publish SNS notification if configured. Returns appropriate HTTP status.

2.3 Bulk Import

1. **Browser**: Parses CSV into task objects; calls POST /tasks/bulk-import with array.
2. **API Gateway → TaskHandler**:
    - Reads config (SQS URL) from Parameter Store.
    - Splits tasks into batches ≤10; sends each batch to SQS.
    - Publishes “bulk import started” SNS message.
    - Returns HTTP 202 Accepted.
3. **SQS Queue**: Messages accumulate.
4. **SQSProcessor Lambda**: Triggered by queue:
    - Processes each batch: for each message, writes to DynamoDB Tasks table.

- On full batch success: publishes “bulk import completed” SNS message.
- On failures: relies on automatic retry or DLQ configuration.

2.4 Analytics

1. **Scheduled Event**: EventBridge triggers Analytics Lambda daily.
2. **Analytics Lambda**:
   - Scans Tasks table (pagination).
   - Computes metrics (e.g., total tasks per day, status counts, priority distribution, average completion).
   - Writes one or more items into Analytics table with PK = date, SK = metric name.
3. **Admin API**: GET /admin/analytics → API Gateway → Analytics Lambda: reads Analytics table (query by date or range) and returns JSON for dashboard.

3. Security and Access Control

- **Cognito JWT**: API Gateway authorizer ensures only authenticated requests reach Lambdas.
- **Groups**: Lambda logic inspects JWT groups claim. Regular users can operate only on their own tasks; admins have broader access (e.g., view all tasks or analytics).
- **IAM Roles**:
  - Lambdas run under roles granting precisely scoped permissions:
    - TaskHandler: DynamoDB (Tasks), SNS publish, SQS send, Parameter Store read.
    - SQSProcessor: DynamoDB (Tasks), SNS publish.
    - Analytics: DynamoDB (scan/write Analytics), Parameter Store read.
    - Cognito triggers: cognito-idp:AdminAddUserToGroup (for AutoAssign).
- **Parameter Store**: Centralizes config (SNS ARN, SQS URL). Could be a SecureString if sensitive, but ARNs/URLs are not secrets.
- **CORS**: API Gateway configured to allow only the SPA origin.
- **Input Validation**: Lambdas validate incoming data (e.g., required fields, date format).
- **Least Privilege**: Avoid wildcard permissions; scope to resource ARNs.
- **CloudTrail & CloudWatch**: Track changes and runtime errors.

4. Deployment & Configuration

- **Infrastructure as Code**: Use CloudFormation (or AWS SAM/CDK) template that:
  - Creates S3 bucket (static hosting), Cognito User Pool & app client (with redirect URIs), DynamoDB tables, SNS topic & subscription, SQS queue, Parameter Store parameter, Lambdas (with code

packages), API Gateway resources and authorizer, EventBridge rule, CloudTrail trail.
    - Outputs necessary values (S3 website URL, Cognito domain & clientId, API endpoint) for frontend config.
- **Frontend Update**: After deployment, update js/config.js with actual Cognito domain/client ID, redirectUri, and API endpoint. Upload updated files to S3.
- **Lambda Code**: Package each function (TaskHandler, SQSProcessor, Analytics, Cognito triggers). Ensure dependencies are included. Configure environment variables or rely on Parameter Store for external config.
- **Parameter Store Initialization**: After SNS and SQS are created, write their ARNs/URLs into /taskflow/config. Lambdas read this parameter at runtime.
- **Testing**:
    - Sign up a user, check pre-signup validation and auto-group assignment.
    - CRUD tasks, verify notifications.
    - Bulk import small CSV, observe queued processing and completion notification.
    - Verify analytics run (either trigger manually or wait for scheduled run), then call admin endpoint.
- **Monitoring**: Set up CloudWatch dashboards/alarms for Lambda errors and SQS queue depth; ensure CloudTrail is logging.

5. Simplified Sequence Overview

Below is a very brief sequence outline; refer to the diagram for arrows and placement.

1. **User Sign-Up**: Browser → Cognito (with PreSignupValidation & AutoAssignGroup Lambdas) → user in users group.
2. **User Authenticated**: Browser holds JWT.
3. **Task CRUD**: Browser (JWT) → API Gateway (Cognito authorizer) → TaskHandler Lambda → DynamoDB (Tasks) → SNS notification → Admin email.
4. **Bulk Import**: Browser → API Gateway → TaskHandler → SQS → SQSProcessor Lambda → DynamoDB → SNS notification.
5. **Analytics**: EventBridge → Analytics Lambda → DynamoDB scan/write → Admin reads via API.
6. **Config Flow**: Lambdas read SNS/SQS config from Parameter Store; update by editing Parameter Store if resources change.
7. **Monitoring/Audit**: CloudWatch collects logs/metrics; CloudTrail logs all API calls.